APOSTILA KIT

ARDUINO INTERMEDIATE

Sumário

Parte I - Revisão de circuitos elétricos e componentes básicos

Introdução

A primeira parte da apostila faz uma revisão sobre circuitos elétricos, com destaque para questões práticas de montagem e segurança que surgem no dia-a-dia do usuário do Kit Arduino INTERMEDIATE.

O conteúdo de circuitos elétricos aborda divisores de tensão e corrente, conversão de níveis lógicos, grandezas analógicas e digitais, níveis de tensão e cuidados práticos de montagem.
Em relação aos componentes básicos, é feita uma breve discussão sobre resistores elétricos, capacitores, leds, diodos, chaves e protoboards.

O Arduino UNO é o principal componente do Kit e é discutido e introduzido em uma seção à parte, na Parte II da apostila.

Todos os conteúdos da Parte I são focados na apostila Arduino INTERMEDIATE, tendo em vista a utilização adequada dos componentes e da realização prática de montagens pelos usuários. No entanto, recomenda-se a leitura das referências indicadas ao final de cada seção para maior aprofundamento.

O leitor veterano, já acostumado e conhecedor dos conceitos essenciais de eletrônica e eletricidade, pode pular a Parte I e ir direto a Parte II, na qual são apresentadas uma seção de exemplo de montagem para cada sensor ou componente importante da apostila.

Algum conhecimento prévio é bem-vindo, mas não é necessário para utilização do kit. Caso seja seu primeiro contato, nós recomendamos que antes de fazer as montagens você leia esses três artigos do http://blog.eletrogate.com/

- https://blog.eletrogate.com/o-que-e-arduino-para-que-serve-vantagens-e-como-utilizar/
- *http://blog.eletrogate.com/programacao-arduino-parte-1/*
- *http://blog.eletrogate.com/programacao-arduino-parte-2/*

Preparado? Vamos começar!

Revisão de circuitos elétricos

A apostila Arduino INTERMEDIATE, bem como todas as outras apostilas que tratam de Arduino e eletrônica em geral, tem como conhecimento de base as teorias de circuitos elétricos e de eletrônica analógica e digital.

Do ponto de vista da teoria de circuitos elétricos, é importante conhecer os conceitos de grandezas elétricas: Tensão, corrente, carga, energia potência elétrica. Em todos os textos sobre Arduino ou qualquer assunto que envolva eletrônica, você sempre terá que lidar com esses termos. Para o leitor que se inicia nessa seara, recomendamos desde já que mesmo que a eletrônica não seja sua área de formação, que conheça esses conceitos básicos.

Vamos começar pela definição de “circuito elétrico”. *Um circuito elétrico/eletrônico é uma interconexão de elementos elétricos/eletrônicos*. Essa interconexão pode ser feita para atender a uma determinada tarefa, como acender uma lâmpada, acionar um motor, dissipar calor em um resistência e tantos outras.

O circuito pode estar **energizado** ou **desenergizado**. Quando está energizado, é quando uma fonte de tensão externa ou interna está ligada aos componentes do circuito. Nesse caso, uma corrente elétrica fluirá entre os condutores do circuito. Quando está desenergizado, a fonte de tensão não está conectada e não há corrente elétrica fluindo entre os condutores.

Mas atenção, alguns elementos básicos de circuitos, como os capacitores ou massas metálicas, são elementos que armazenam energia elétrica. Em alguns casos, mesmo não havendo fonte de tensão conectada a um circuito, pode ser que um elemento que tenha energia armazenada descarregue essa energia dando origem a uma corrente elétrica transitória no circuito. Evitar que elementos do circuito fiquem energizados mesmo sem uma fonte de tensão, o que pode provocar descargas elétricas posteriores (e em alguns casos, danificar o circuito ou causar choques elétricos) é um dos motivos dos sistemas de aterramento em equipamentos como osciloscópios e em instalações residenciais , por exemplo.

Em todo circuito você vai ouvir falar das grandezas elétricas principais, assim, vamos aprender o que é cada uma delas.

Carga e corrente elétrica

A grandeza mais básica nos circuitos elétricos é a carga elétrica. Carga é a propriedade elétrica das partículas atômicas que compõem a matéria, e é medida em Coulombs. Sabemos da física elementar que a matéria é constituída de elétrons, prótons e neutros. **A carga elementar é a carga de 1 elétron, que é igual a 1,602 x10-19 C**.

Do conceito de carga elétrica advém o **conceito de corrente elétrica, que nada mais é do que a taxa de variação da carga em relação ao tempo**, ou seja, quando você tem um fluxo de carga em um condutor, a quantidade de carga (Coulomb) que atravessa esse condutor por unidade de tempo, é chamada de corrente elétrica. A medida utilizada para corrente é o **Ampére(A).**

Aqui temos que fazer uma distinção importante. Existem corrente elétrica contínua e alternada:

- **Corrente elétrica contínua:** É uma corrente que permanece constante e em uma única direção durante todo o tempo.
- **Corrente elétrica alternada**: É uma corrente que varia senoidalmente (ou de outra forma) com o tempo.

Com o Arduino UNO, lidamos como correntes elétricas contínuas, pois elas fluem sempre em uma mesma direção. É diferente da corrente e tensão elétrica da tomada de sua casa, que são alternadas.

Ou seja, os seus circuitos com Arduino UNO sempre serão alimentados com grandezas contínuas (corrente e tensão contínuas).

Tensão elétrica

Para que haja corrente elétrica em um condutor, é preciso que os elétrons se movimentem por ele em um determinada direção, ou seja, é necessário “alguém” para transferir energia para as cargas elétricas para movê-las. Isso é feito por uma força chamada **força eletromotriz (fem),** tipicamente representada por uma bateria. Outros dois nomes comuns para força eletromotriz são **tensão elétrica** e **diferença de potencial.**

O mais comum é você ver apenas "*tensão*" nos artigos e exemplos com Arduino. Assim, definindo formalmente o conceito: Tensão elétrica é a energia necessária para mover uma unidade de carga através de um elemento, e é medida em **Volts (V).**

Potência e energia

A tensão e a corrente elétrica são duas grandezas básicas, e juntamente com a potência e energia, são as grandezas que descrevem qualquer circuito elétrico ou eletrônico. A potência é definida como a variação de energia (que pode estar sendo liberada ou consumida) em função do tempo, e é medida em **Watts (W).** A potência está associada ao calor que um componente está dissipando e a energia que ele consume.

Nós sabemos da vida prática que uma lâmpada de 100W consome mais energia do que uma de 60 W. Ou seja, se ambas estiverem ligadas por 1 hora por exemplo, a lâmpada de 100W vai implicar numa conta de energia mais cara.
A potência se relaciona com a tensão e corrente pela seguinte equação:

$$P = V \times I$$

Essa é a chamada potência instantânea. Com essa equação você saber qual a potência dissipada em um resistor por exemplo, bastando que saiba a tensão nos terminais do resistor e a corrente que flui por ele. O conceito de potência é importante pois muitos hobbystas acabam não tendo noção de quais valores de resistência usar, ou mesmo saber especificar componentes de forma adequada.

Um resistor de 33 ohms de potência igual a 1/4W, por exemplo, não pode ser ligado diretamente em circuito de 5V, pois nesse caso a potência dissipada nele seria maior que a que ele pode suportar.

Vamos voltar a esse assunto em breve, por ora, tenha em mente que é importante ter uma noção da potência dissipada ou consumida pelos elementos de um circuito que você vá montar.

Por fim, a energia elétrica é o somatório da potência elétrica durante todo o tempo em que o circuito esteve em funcionamento. A energia é dada em **Joules (J)** ou **Wh (watt-hora).** A unidade Wh é interessante pois mostra que a energia é calculada multiplicando-se a potência pelo tempo(apenas para os casos em que a potência é constante).

Essa primeira parte é um pouco conceitual, mas é importante saber de onde vieram toda a terminologia que você sempre vai ler nos manuais e artigos na internet. Na próxima seção, vamos discutir os componentes básicos de circuito que compõem o Kit Arduino INTERMEDIATE.

Conversão de níveis lógicos e alimentação correta de circuitos

É muito comum que hobbystas e projetistas em geral acabem por cometer alguns erros de vez em quando. Na verdade, mesmo alguns artigos na internet e montagens amplamente usadas muitas vezes acabam por não utilizar as melhores práticas de forma rigorosa. Isso acontece muito no caso dos níveis lógicos dos sinais usados para interfacear o Arduino com outros circuitos.

Como veremos na seção de apresentação do Arduino UNO, o mesmo é alimentado por um cabo USB ou uma fonte externa entre 6V e 12V. O Circuito do Arduino possui reguladores de tensão que convertem a alimentação de entrada para 5V e para 3V. Os sinais lógicos das portas de saída(I/Os) do Arduino, variam entre 0 e 5V.

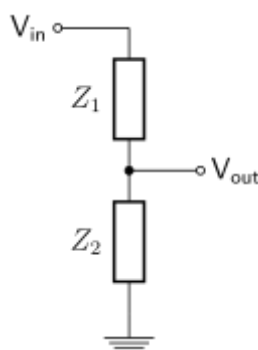
Isso significa que quando você quiser usar o seu Arduino UNO com um sensor ou CI que trabalhe com 3.3V, é preciso fazer a adequação dos níveis de tensão, pois se você enviar um sinal de 5V (saída do Arduino) em um circuito de 3.3V (CI ou sensor), você poderá queimar o pino daquele componente.
Em geral, sempre que dois circuitos que trabalhem com níveis de tensão diferentes forem conectados, é preciso fazer a conversão dos níveis lógicos. O mais comum é ter que abaixar saídas de 5V para 3.3V. Subir os sinais de 3.3V para 5V na maioria das vezes não é necessário pois o Arduino entende 3.3V como nível lógico alto, isto é, equivalente a 5V.

Para fazer a conversão de níveis lógicos você tem duas opções:

- Usar um divisor de tensão;
- Usar um CI conversor de níveis lógicos;

O divisor de tensão é a solução mais simples, mas usar um CI conversor é mais elegante e é o ideal. O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles. O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Em que Z1 e Z2 são os valores dos resistores da figura abaixo.

Um divisor de tensão muito comum é fazer Z1 igual 330 ohms e Z2 igual 680 ohms. Dessa forma a sáida Vout fica sendo 3.336 V. Como no Kit Arduino INTERMEDIATE não há resistor de 680ohms, você pode ligar um de 330ohms como Z1 e dois de 330ohms como Z2. Os dois de 330 ligados em série formam um resistor de 660 ohm, o que resulta numa saída de 3.33V.

Vamos exemplificar como fazer um divisor de tensão como esse na seção de exemplos da parte II da apostila.

Revisão de componentes eletrônicos

O Kit Arduino INTERMEDIATE possui os seguintes componentes básicos para montagens de circuitos:

- Buzzer Ativo 5V,
- LED Vermelho/ Verde/ Amarelo,
- Resistor 330Ω/ 1KΩ/ 10KΩ,
- Diodo 1N4007,
- Potenciômetro 10KΩ,
- Capacitor Cerâmico 10 nF/ 100 nF,
- Capacitor Eletrolítico 10uF/ 100uF,
- Chave Táctil (Push-Button).

Vamos revisar a função de cada um deles dentro de um circuito eletrônica e apresentar mais algumas equações fundamentais para ter em mente ao fazer suas montagens.

Resistores elétricos

Os resistores são componentes que se opõem à passagem de corrente elétrica, ou seja, oferecem uma resistência elétrica. Dessa forma, quanto maior for o valor de um resistor, menor será a corrente elétrica que fluirá por ele e pelo condutor a ele conectada. A unidade de resistência elétrica é o **Ohm(Ω)**, que é a unidade usada para especificar o valor dos resistores.

Os resistores mais comuns do mercado são construídos com fio de carbono e são vendidos em várias especificações. Os resistores do Kit são os tradicionais de 1/4W e 10% de tolerância. Isso significa que eles podem dissipar no máximo 1/4W (0,25 watts) e seu valor de resistência pode variar em até 10%. O resistor de 1KΩ pode então ter um valor mínimo de 900Ω e um valor máximo de 1100Ω.

Em algumas aplicações você pode precisar de resistores com precisão maior, como 5% ou 1%. Em outros casos, pode precisar de resistores com potência maior, como 1/2W ou menor, como 1/8W. Essas variações dependem da natureza de cada circuito.

Em geral, para as aplicações típicas e montagens de prototipagem que podem ser feitos com o Kit Arduino INTERMEDIATE, os resistores tradicionais de 1/4W e 10% de tolerância são suficientes.

Outro ponto importante de se mencionar aqui é a Lei de Ohm, que relaciona as grandezas de tensão, corrente e resistência elétrica. A lei é dada por:

$$V = R \times I$$

Ou seja, se você sabe o valor de um resistor e a tensão aplicada nos seus terminais, você pode calcular a corrente elétrica que fluirá por ele. Juntamente com a equação para calcular potência elétrica, a lei de Ohm é importante para saber se os valores de corrente e potência que os resistores de seu circuito estão operando estão adequados.

Para fechar, você deve estar se perguntando, como saber o valor de resistência de um resistor? Você tem duas alternativas: Medir a resistência usando um multímetro ou determinar o valor por meio do código de cores do resistor.

Se você pegar um dos resistores do seu kit, verá que ele possui algumas faixas coloridas em seu corpo. Essas faixas são o código de cores do resistor. As duas primeiras faixas dizem os dois primeiros algarismos decimais. A terceira faixa colorida indica o multiplicador que devemos usar. E a última faixa, que fica um pouco mais afastada, indica a tolerância.

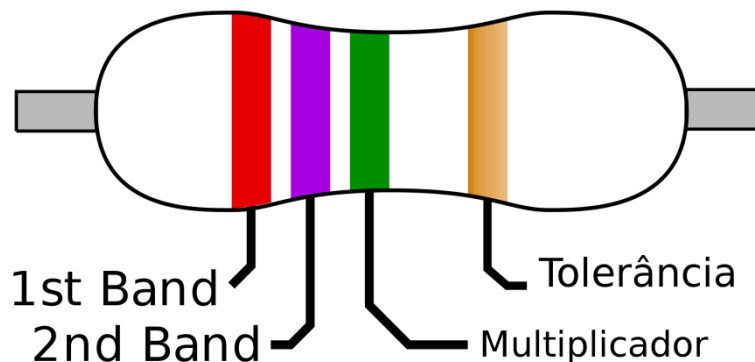



Figura 1: Faixas coloridas em um resistor

Na figura 2 apresentamos o código de cores para resistores. Cada cor está associada a um algarismo, um multiplicador e uma tolerância, conforme a tabela. Com a tabela você pode determinar a resistência de um resistor sem ter que usar o multímetro.

Mas atenção, fazer medições com o multímetro é recomendado, principalmente se o componente já tiver sido utilizado, pois o mesmo pode ter sofrido algum dano ou mudança que não esteja visível.

Cor	Dígito	Multiplicador	Tolerância
Prata	-	x 0,01	± 10%
Dourado	-	x 0,1	± 5%
Preto	0	x 1	-
Marrom	1	x 10	± 1%
Vermelho	2	x 100	± 2%
Laranja	3	x 1K	-
Amarelo	4	x 10K	-
Verde	5	x 100K	± 0,5%
Azul	6	x 1M	± 0,25%
Violeta	7	x 10M	± 0,1%
Cinza	8	-	± 0,05%
Branco	9	-	-

Figura 2: Código de cores para resistores

Aplicando a tabela da figura 2 na imagem da figura 1, descobrimos que o resistor é de 2,7MΩ (Mega ohms) com tolerância de 5% (relativo à cor dourado da última tira).

Capacitores

Os capacitores são os elementos mais comuns nos circuito eletrônicos depois dos resistores. São elementos que armazenam energia na forma de campos elétricos. Um capacitor é constituído de dois terminais condutores e um elemento dielétrico entre esses dois terminais, de forma que quando submetido a uma diferença de potencial, um campo elétrico surge entre esses terminais, causando o acúmulo de cargas positivas no terminal negativo e cargas negativas no terminal positivo.

São usados para implementar filtros, estabilizar sinais de tensão, na construção de fontes retificadores e várias outras aplicações.

O importante que você deve saber para utilizar o Kit é que os capacitores podem ser de quatro tipos:

- Eletrolíticos,
- Cerâmicos,
- Poliéster,
- Tântalo.

Capacitor eletrolítico

As diferenças de cada tipo de capacitor são a tecnologia construtiva e o material dielétrico utilizado. Capacitores eletrolíticos são feitos de duas longas camadas de alumínio (terminais) separadas por uma camada de óxido de alumínio (dielétrico). Devido a sua construção, eles possuem **polaridade,** o que significa que você obrigatoriamente deve ligar o terminal positivo (quem tem uma “perninha” maior) no pólo positivo da tensão de alimentação, e o terminal negativo (discriminado no capacitor por uma faixa com símbolos de “-”) obrigatoriamente no pólo negativo da bateria. Do contrário, o capacitor será danificado.

Capacitores eletrolíticos costumam ser da ordem de micro Farad, sendo o **Farad** a unidade de medida de capacitância, usada diferenciar um capacitor do outro. A Figura 3 ilustra um típico capacitor eletrolítico.

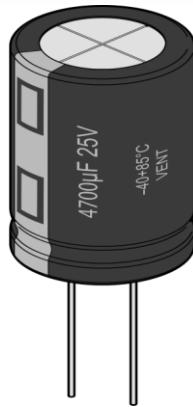



Figura 3: Capacitor eletrolítico 4700 micro Farads / 25 V

Capacitores cerâmicos

Capacitores cerâmicos não possuem polaridade, e são caracterizados por seu tamanho reduzido e por sua cor característica, um marrom claro um tanto fosco. Possuem capacitância da ordem de **pico Farad**. Veja nas imagens abaixo um típico capacitor cerâmico e sua identificação:

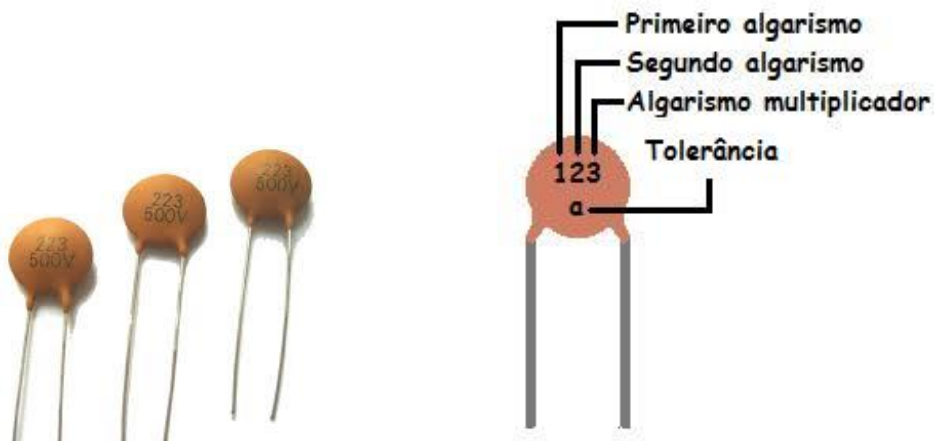



Figura 4: Capacitores cerâmicos

Na imagem da esquerda, os capacitores possuem o valor de **22 nano Farads** para a faixa de tensão de até 500V.

223 = 22 x 1000 = 22.000 pF = 22 nF

Por fim, há também os capacitores de poliéster e de tântalo. No Kit Arduino INTERMEDIATE, você receberá apenas exemplares de capacitores eletrolíticos e cerâmicos.

Diodos e Leds

Diodos e Leds são tratados ao mesmo tempo pois tratam na verdade do mesmo componente. Diodos são elementos semicondutores que só permitem a passagem de corrente elétrica em uma direção.

São constituídos de dois terminais, o **Anodo(+)** e o **catodo(-)**, sendo que para que possa conduzir corrente elétrica, é preciso conectar o Anodo na parte positiva do circuito, e o Catodo na parte negativa. Do contrário, o diodo se comporta como um circuito aberto.

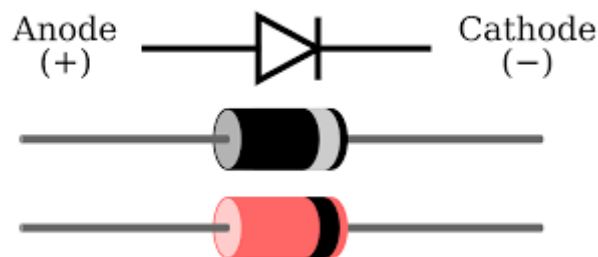



Figura 4: Diodo e seus terminais

Na figura 4, você pode ver que o componente possui uma faixa indicadora do terminal de catodo. O diodo do kit Arduino INTERMEDIATE é um modelo tradicional e que está no mercado há muitos anos, o 1N4007.

O LED é um tipo específico de diodo - **Light Emitter Diode**, ou seja, diodo emissor de luz. Trata-se de um diodo que quando polarizado corretamente, emite luz para o ambiente externo. O Kit Arduino BEGINNIG vem acompanhado de leds na cor vermelha, verde e amarelo, as mais tradicionais.

Nesse ponto, é importante você saber que sempre deve ligar um led junto de um resistor, para que a corrente elétrica que flua pelo led não seja excessiva e acabe por queimá-lo. Além disso, lembre-se que por ser um diodo, o led só funciona se o Anodo estiver conectado ao pólo positivo do sinal de tensão.

Para identificar o Anodo do Led, basta identificar a perna mais longa do componente, como na imagem abaixo:

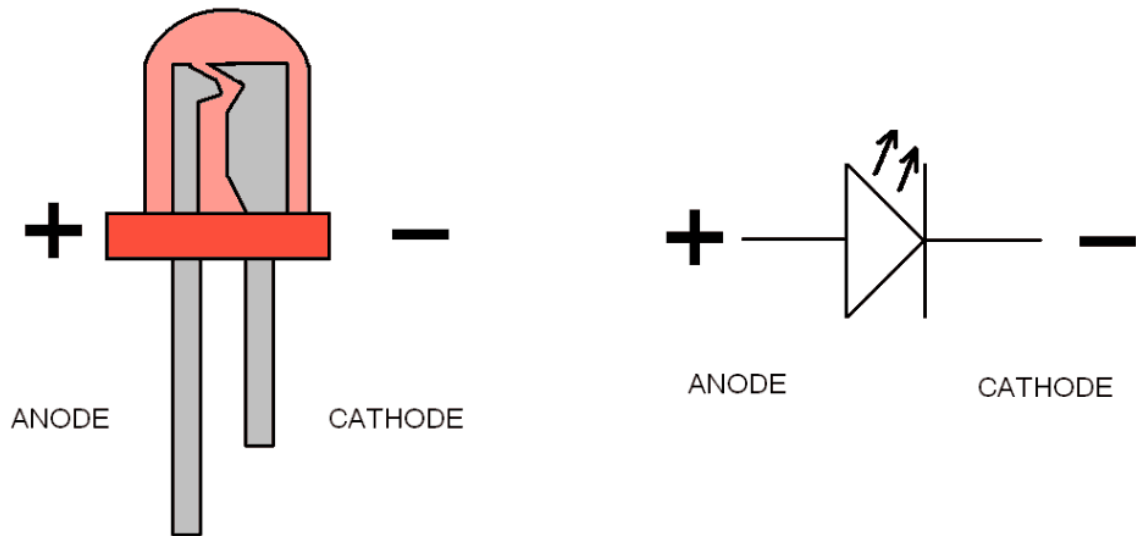



Figura 5: Terminais de um Led. Créditos: Build-eletronic-circuits.com

Os demais componentes da apostila, Buzzer, potenciômetro e push-buttons, serão explicados em seções de exemplos, nas quais iremos apresentar um circuito prático para montagem onde será explicado o funcionamento dos mesmos.

Os sensores do Kit Arduino INTERMEDIATE, quais sejam: Sensor de Luz LDR e sensor de temperatura NTC, juntamente com o Micro Servo 9g SG90 TowerPro, também terão uma seção de exemplo dedicada a cada um deles.

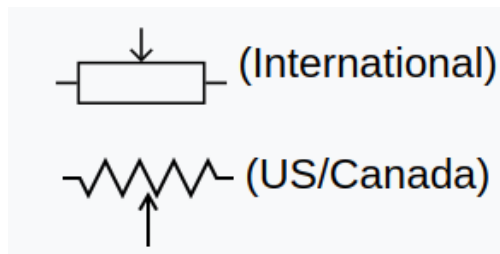
Parte II - Seção de Exemplos Práticos

Agora vamos entrar nas seções de exemplos em si. Os conceitos da Parte I são importantes caso você esteja começando a trabalhar com eletrônica. No entanto, devido ao aspecto prático da montagem, não necessariamente você precisa de ler toda a parte introdutória para montar os circuitos abaixo.

Em cada exemplo vamos apresentar os materiais utilizados, o diagrama de circuito e o código base com as devidas explicações e sugestões de referências.

Exemplo 1 - Usando potenciômetro para fazer leituras analógicas

O potenciômetro nada mais do que um resistor cujo valor de resistência pode ser ajustado de forma manual. Existem potenciômetros slides e giratórios. Na figura abaixo mostramos um potenciômetro giratório dos mais comumente encontrados no mercado.

Em ambos, ao variar a posição da chave manual, seja ela giratória ou slide, o valor da resistência entre o terminal de saída e um dos outros terminais se altera. O símbolo do potenciômetro é o mostrado na seguinte imagem:

Nesse exemplo, vamos ligar a saída de um potenciômetro a uma entrada analógica da Arduino UNO. Dessa forma, vamos ler o valor de tensão na saída do potenciômetro e vê-la variando de
0 a 1023. Mas como assim, 0 a 1023?

Isso se deve ao seguinte. Vamos aplicar uma tensão de 5V nos terminais do potenciômetro. A entrada analógica do Arduino, ao receber um sinal de tensão externo, faz a conversão do mesmo para um valor digital, que é representado por um número inteiro de 0 a 1023. Esses números são assim devido à quantidade de bits que o conversor analógico digital do Arduino trabalha, que são 10 bits (2 elevado a 10 = 1024).

Ou seja, o arduino divide o valor de tensão de referência em 1024 unidades (0 a 1023) de 0,00488 volts. Assim, se a tensão lida na entrada analógica for de 2,5V, o valor capturado pelo arduino será metade de 2,5/0,00488 = 512. Se for 0V, será 0, e ser for 5V, será 1023, e assim proporcionalmente para todos os valores. Assim, digamos que o valor de tensão se dado por V. O valor que o Arduino vai te mostrar é:

Valor = (V/5)*1024

Em que 5V é o valor de referência configurado no conversor analógico-digital (uma configuração já padrão, não se preocupe com ela) e 1024 é igual 2 elevado a 10.

No nosso código, queremos saber o valor de tensão na saída do potenciômetro, e não esse número de 0 a 1023, assim, rearranjar a equação para o seguinte:

Tensão = Valor*(5/1024)

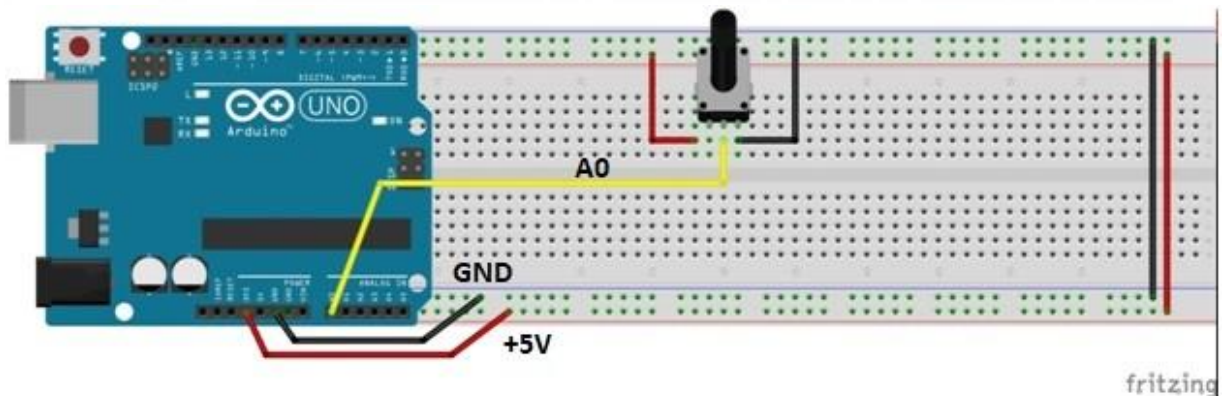
Bacana, né? Agora, vamos à montagem em si.

Lista de materiais:

Para esse exemplo você vai precisar:

- Arduino UNO;
- Protoboard;
- Potenciômetro 10K;
- Jumpers de ligação;

Diagrama de circuito

Monte o circuito conforme diagrama abaixo e carregue o código de exemplo:

O Potenciômetro possui 3 terminais, sendo que o do meio é o que possui resistência variável. A ligação consiste em ligar os dois terminais fixos a um tensão de 5V. Assim, o terminal intermediário do potenciômetro terá um valor que varia de 0 a 5V à medida que você dá gira seu knob.

O terminal intermediário é ligado diretamente a uma entrada analógica do Arduino (A0). Como a tensão é de no máximo 5V, então não há problema em ligar direto.

Carregue o código abaixo no Arduino e você verá as leituras no Monitor serial da IDE arduino.

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT INTERMEDIATE

#define sensorPin A0      // define entrada analógica A0

int sensorValue = 0;      // variável inteiro igual a zero
float voltage;            // variável numero fracionario

void setup()
{
  Serial.begin(9600);     // monitor serial - velocidade 9600 Bps
  delay(100);             // atraso de 100 milisegundos
}

void loop()
{
  sensorValue = analogRead(sensorPin);        // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024);       // cálculo da tensão

  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage);                      // imprime a tensão
  Serial.print("    Valor: ");                // imprime no monitor serial
  Serial.println(sensorValue);                // imprime o valor
  delay(500);                                 // atraso de 500 milisegundos
}
```

No código acima, nós declaramos a variável **sensorValue** para armazenar as leituras da entrada analógica A0 e a variável do tipo **float Voltage** para armazenar o valor lido convertido para tensão.

Na função **void Setup()**, nós inicializamos o terminal serial com uma taxa de transmissão de 9600 kbps. Na função **void Loop(),** primeiro faz-se a leitura da entrada analógica A0 com a função **analogRead(SensorPin)** e armazenamos a mesma na variável **sensorValue**. Em seguida, aplicamos a fórmula para converter a leitura (que é um número entre 0 e 1023) para o valor de tensão correspondente. O resultado é armazenado na variável **Voltage** e em seguido mostrado na interface serial da IDE Arduino.

Referência:

1. https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Exemplo 2 - Divisor de tensão

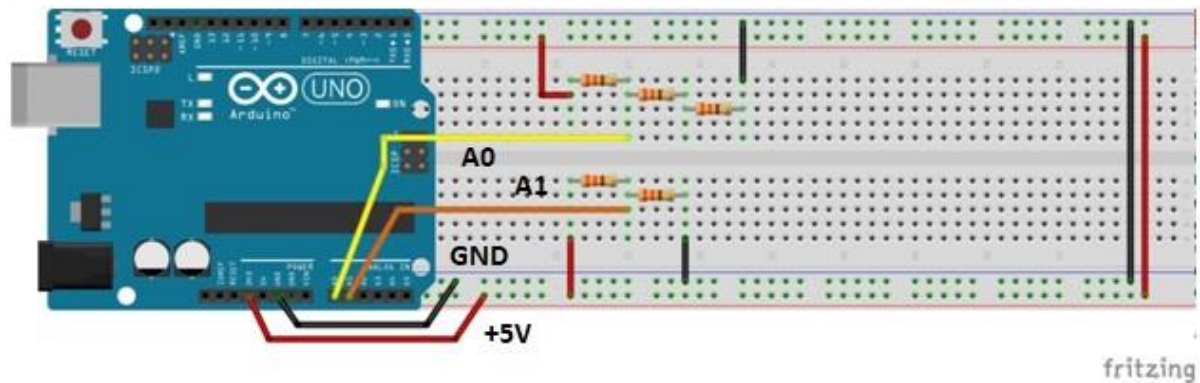
Esse exemplo é para ilustrar como usar um divisor de tensão. Sempre que você precisar abaixar um sinal lógico de 5V para 3.3V você pode usar esse circuito. Explicamos o divisor de tensão na seção de introdução, mais especificamente, quando conversamos sobre conversão de níveis lógicos.

Esse circuito será útil sempre que você tiver que abaixar as saídas do Arduino de 5V para 3.3V.

Lista de materiais:

Para esse exemplo você vai precisar:

- 3 resistores de 330 ohms;
- 2 resistores de 1K ohm;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama de circuito:

Esse é um diagrama com dois divisores de tensão. O primeiro é composto por três resistores, sendo cada um de 330. Assim, o resistor Z1 é de 330 e o resistor Z2 é a associação série dos outros dois, resultando numa resistência de 660. Dessa forma, a tensão de saída do divisor é:

(660 / 330 + 660) * 5 = 3,33V

O segundo divisor é formado por dois resistores de 1K, dessa forma, a tensão de saída é a tensão de entrada dividida pela metade:

(1000 / 1000 + 1000) * 5 = 2,5 V

O código para o exemplo 2 é uma extensão do código usada na seção anterior para ler valores de tensão do potenciômetro. Nesse caso, nós fazemos a leitura de dois canais analógicos (A0 e A1), fazemos as conversões para as tensões e mostramos os resultados de cada divisor na interface serial.

Referências:

1. https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage
2. http://www.ufjf.br/fisica/files/2013/10/FIII-05-07-Divisor-de-tens%C3%A3o.pdf
3. http://www.sofisica.com.br/conteudos/Eletromagnetismo/Eletrodinamica/associacaoderesistores.php

Carregue o código abaixo e observe os dois valores no Monitor Serial da IDE Arduino.

```
// Exemplo 2 - Divisor de tensão
// Apostila Eletrogate - KIT INTERMEDIATE

#define sensorPin1 A0           // define entrada analógica A0
#define sensorPin2 A1           // define entrada analógica A1

int sensorValue1 = 0;           // variavel inteiro igual a zero
int sensorValue2 = 0;           // variavel inteiro igual a zero
float voltage1;                 // variavel numero fracionario
float voltage2;                 // variavel numero fracionario

void setup()
{
  Serial.begin(9600);           // monitor serial - velocidade 9600 Bps
  delay(100);                   // atraso de 100 milisegundos
}

void loop()
{
  sensorValue1 = analogRead(sensorPin1);   // leitura da entrada analógica A0
  sensorValue2 = analogRead(sensorPin2);   // leitura da entrada analógica A1
  voltage1 = sensorValue1 * (5.0 / 1024);  // cálculo da tensão 1
  voltage2 = sensorValue2 * (5.0 / 1024);  // cálculo da tensão 2
  Serial.print("Tensao do divisor 1: ");   // imprime no monitor serial
  Serial.print(voltage1);                  // imprime a tensão 1
  Serial.print("  Tensao do divisor 2: "); // imprime no monitor serial
  Serial.println(voltage2);                // imprime a tensão 2
  delay(500);                              // atraso de 500 milisegundos
}
```

Exemplo 3 - Entradas e saídas digitais - push-button + led

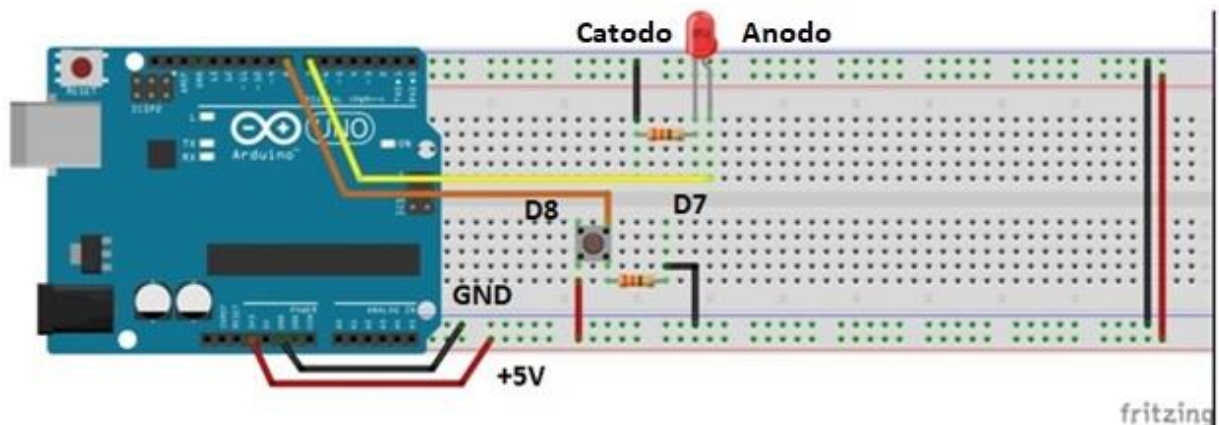
Os push-buttons (chaves botão) e leds são elementos presentes em praticamente qualquer circuito eletrônico. As chaves são usadas para enviar comandos para o Arduino e os Leds são elementos de sinalização luminosa.

Esses dois componentes são trabalhados por meio das entradas e saídas digitais do Arduino. Neste exemplo vamos fazer uma aplicação básica que você provavelmente vai repetir muitas vezes. Vamos ler o estado de um push-button e usá-la para acender ou apagar um led. Ou seja, sempre que o botão for acionado, vamos apagar ou acender o Led.

Lista de materiais:

Para esse exemplo você vai precisar:

- 2 resistores de 330 ohms;
- 1 Led vermelho (ou de outra cor de sua preferência);
- Push-button (chave botão);
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama de circuito:

Esse pequeno código abaixo mostra como ler entradas digitais e com acionar as saídas digitais. Na função **void Setup(),** é preciso configurar qual pino será usado como saída e qual será usado como entrada.

Depois de configurar os pinos, para acioná-los basta chamar a função **digitalWrite(pino,HIGH).** A função **digitalWrite()** aciona ou desaciona um pino digital dependendo do valor passado no argumento. Se for "HIGH", o pino é acionado. Se for "LOW", o pino é desligado.

Na função **void Loop(),** fizemos um if no qual a função **digitalRead** é usada para saber se o pushButton está acionado ou não. Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

Carregue o código abaixo e pressione o botão para acender o LED.

```
// Exemplo 3 - Entradas e saídas digitais - push-button + led
// Apostila Eletrogate - KIT INTERMEDIATE

#define PinButton 8                    // define pino digital D8
#define ledPin 7                       // define pino digital D7

void setup()
{
  pinMode(PinButton, INPUT);           // configura D8 como entrada digital
  pinMode(ledPin, OUTPUT);             // configura D7 como saída digital
  Serial.begin(9600);                  // monitor serial - velocidade 9600 Bps
  delay(100);                          // atraso de 100 milisegundos
}

void loop()
{
  if ( digitalRead(PinButton) == HIGH)  // se chave = nivel alto
  {
    digitalWrite(ledPin, HIGH);         // liga LED com 5V
    Serial.print("Acendendo Led");      // imprime no monitor serial
  }
  else                                  // senão chave = nivel baixo
  {
    digitalWrite(ledPin, LOW);          // desliga LED com 0V
    Serial.print("Desligando led");     // imprime no monitor serial
  }
  delay(100);                           // atraso de 100 milisegundos
}
```

Referências:

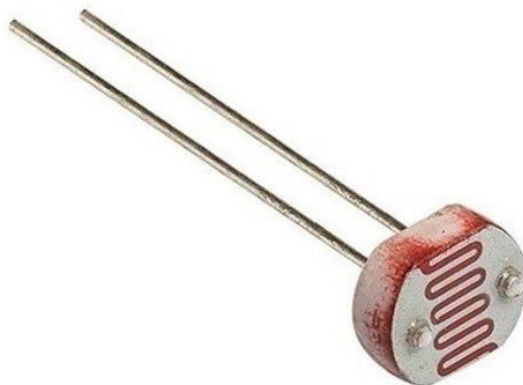
- https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/
- https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

Exemplo 4 - Sensor de luz LDR

O sensor LDR é um sensor de luminosidade. LDR é um **Light Dependent Resistor**, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele. Esse é seu princípio de funcionamento.

Quanto maior a luminosidade em um ambiente, menor a resistência do LDR. Essa variação na resistência é medida através da queda de tensão no sensor, que varia

proporcionalmente (de acordo com a lei Ohm, lembra?) com a queda na resistência elétrica.

A imagem abaixo mostra o sensor em mais detalhes:

Fotoresistor (LDR)

É importante considerar a potência máxima do sensor, que é de 100 mW. Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA. Felizmente, com 8K ohms (que medimos experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA. Dessa forma, podemos interfacear o sensor diretamente com o Arduino.

**Nota: Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.*

Especificações do LDR:

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW
- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)

Este sensor de luminosidade pode ser utilizado em projetos com arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

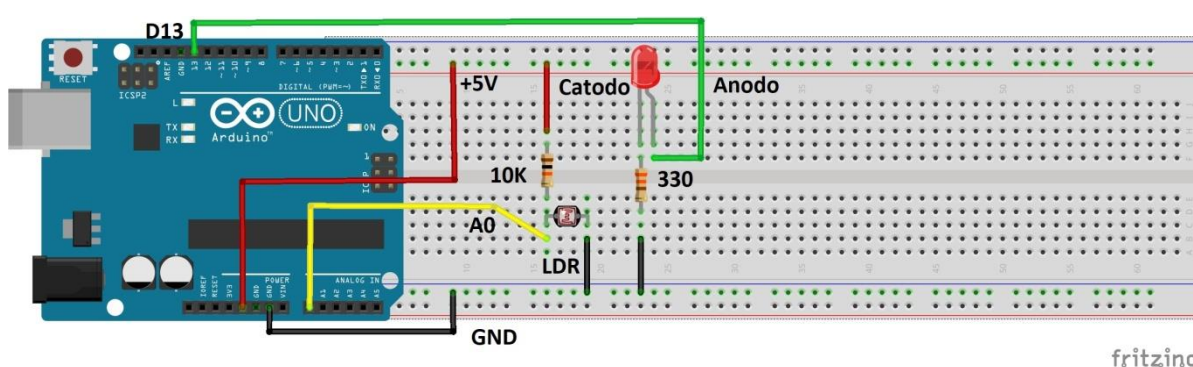
Nesse exemplo, vamos usar uma entrada analógica do Arduino para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando. Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto que no escuro pode chegar a 1MΩ.

Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão. Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos. No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V. Assim, o sinal que vamos ler no arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Lista de materiais:

Para esse exemplo você vai precisar:

- LDR;
- Resistor de 10k;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama de circuito:

No diagrama, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor. Assim, vamos identificar as variações na intensidade de luz pelas variações na tensão do sensor.

Quanto maior a intensidade de luz, menor a resistência do sensor e, consequentemente, menor a tensão de saída.

Carregue o código abaixo e varie a luminosidade sobre o LDR.

```
// Exemplo 4 - Sensor de luz LDR
// Apostila Eletrogate - KIT INTERMEDIATE

#define AnalogLDR A0                    // define pino analógico A0
#define Limiar 1.5                      // define constante igual a 1.5
#define ledPin 13                       // define pino digital D13

int Leitura = 0;                        // variavel inteiro igual a zero
float VoltageLDR;                       // variavel numero fracionario
float ResLDR;                           // variavel numero fracionario

void setup()
{
  pinMode(ledPin, OUTPUT);                  // configura D13 como saída digital
  Serial.begin(9600);                       // monitor serial - velocidade 9600 Bps
  delay(100);                               // atraso de 100 milisegundos
}

void loop()
{
  Leitura = analogRead(AnalogLDR);          // leitura da tensão no pino analogico A0
  VoltageLDR =  Leitura * (5.0/1024);       // calculo da tensão no LDR
  Serial.print("Leitura sensor LDR = ");    // imprime no monitor serial
  Serial.println(VoltageLDR);               // imprime a tensão do LDR

  if (VoltageLDR > Limiar)                  // se a tensão LDR maior do que limiar
    digitalWrite(ledPin, HIGH);             // liga LED com 5V
  else                                      // senão a tensão LDR < limiar
    digitalWrite(ledPin, LOW);              // desliga LED com 0V
  delay(500);                               // atraso de 500 milisegundos
}
```

No código acima usamos funcionalidades de todos os exemplos anteriores. Como o sensor é um elemento de um divisor de tensão, fazemos a sua leitura do mesmo modo que nos exemplos do potenciômetro e divisor de tensão.

Nesse caso, definimos uma tensão de limiar, a partir da qual desligamos ou ligamos um led para indicar que a intensidade da luz ultrapassou determinado valor. Esse limiar pode ser ajustado por você para desligar o led em intensidades diferentes de luz ambiente.

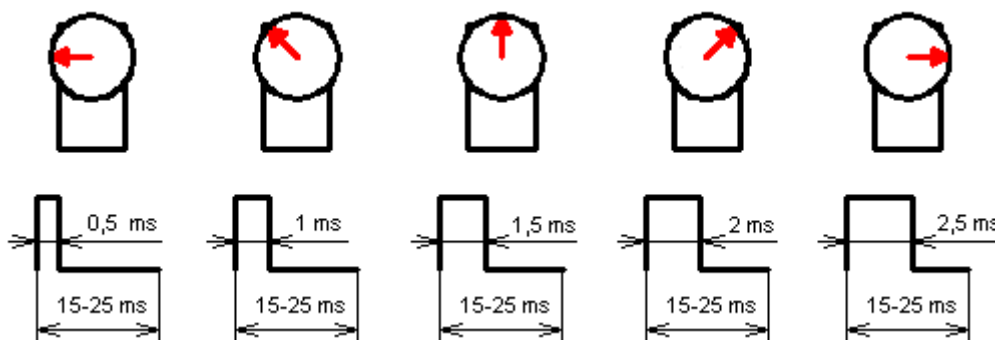
É importante que o sensor esteja exposto à iluminação ambiente e não sofra interferência de fontes luminosas próximas, mas que não sejam parte do ambiente.

Referências:

- https://maker.pro/education/using-an-ldr-sensor-with-arduino-a-tutorial-for-beginners
- http://blog.eletrogate.com/controle-de-luminosidade-com-arduino-e-sensor-ldr/

Exemplo 5 - Acionando o Micro Servo 9g SG90 TowerPro

Um servomotor é um equipamento eletromecânico que possui um encoder e um controlador acoplado. Diferentemente de motores tradicionais, como de corrente contínua, o servo motor apresenta movimento rotativo proporcional a um comando de forma a atualizar sua posição. Ao invés de girar continuamente como os motores de corrente contínua, o servo, ao receber um comando, gira até a posição especificada pelo mesmo.

Ou seja, o servo motor é um atuador rotativo para controle de posição, que atua com precisão e velocidade controlada em malha fechada. De acordo com a largura do pulso aplicado no pino de controle PWM, a posição do rotor é definida (0 a 180 graus) . Os pulsos devem variar entre 0,5 ms e 2,5 ms.

Posição do Servo de acordo com a largura do pulso
Fonte : electronics.stackexchange.com

Existem dois tipos de Servomotores :

- Servomotor analógico
- Servomotor digital

Servomotores analógicos são os mais comuns e mais baratos. O controle da posição do rotor utiliza um método analógico através da leitura de tensão sobre um potenciômetro interno.

No caso dos servomotores digitais, mais caros e mais precisos, o controle da posição do rotor utiliza um encoder digital.

A figura abaixo mostra um típico **servo motor analógico**. Trata-se do **Micro Servo Tower Pro SG90 9G** que acompanha o kit Arduino INTERMEDIATE.

Os Servos são acionados por meio de três fios, dois para alimentação e um correspondente ao sinal de controle para determinar a posição. Em geral, os três fios são:

- Marron: GND,
- Vermelho: Alimentação positiva,
- Laranja: Sinal de controle PWM.

Lista de materiais:

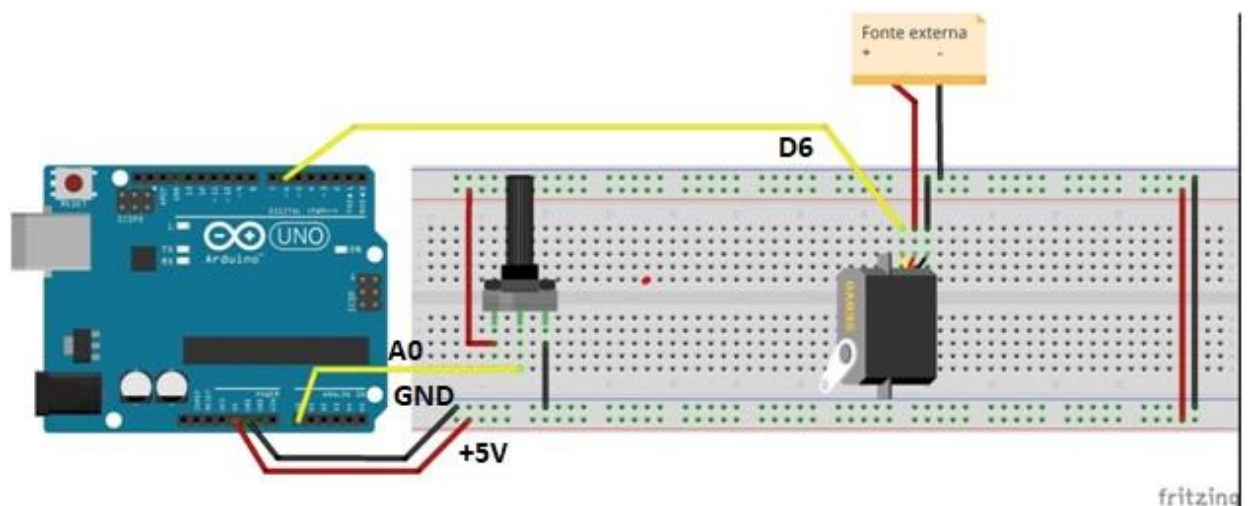
Antes de mais nada, vamos separar os componentes que precisamos para montar o servomotor junto com o arduino. A nossa lista de componentes é a seguinte:

- Micro Servo 9g SG90 TowerPro;
- Arduino UNO + cabo USB;
- Potenciômetro de 10k;

- Fonte 5V de protoboard para alimentar o servo;
- Jumpers para conexão no protoboard;
- Push button;

A montagem é simples. O servomotor em si deve ser ligado à alimentação conforme as cores apresentadas na introdução. De acordo com as especificações de tensão e corrente dos pinos do arduino UNO, os pinos de VCC e GND da placa conseguem fornecer até 200 mA. Ao utilizar servomotores, é recomendado que você utilize uma fonte externa, como a fonte para protoboard que inserimos na lista de materiais.

Diagrama de circuito:

A montagem para uma aplicação de controle do servo em qualquer posição, fica da seguinte forma:

Carregue o código abaixo e gire o potenciômetro para o Servo motor girar:

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT INTERMEDIATE

#include <Servo.h>                     // usando bilbioteca Servo

Servo myservo;                         // cria o objeto myservo

#define potpin A0                      // define pino analógico A0

int val;                               // variavel inteiro

void setup()
{
  myservo.attach(6);                   // configura pino D6 - controle do Servo
}

void loop()
{
  val = analogRead(potpin);            // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179);     // converte a leitura em números (0-179)
  myservo.write(val);                  // controle PWM do servo
  delay(15);                           // atraso de 15 milisegundos
}
```

O aspecto mais importante desse software é a utilização da biblioteca **servo.h**. Esta biblioteca possui as funções necessárias para posicionar a servo para a posição desejada. Na função **Void Setup()** nós associamos o objeto servomotor, do tipo Servo, a um pino do arduino. Na mesma função, nós inicializamos o servo na posição 0º, utilizando o método **write** do objeto que acabamos de criar.

Na função **Void Loop(),** o procedimento consiste em ler o valor do potenciômetro e usá-lo como referência para atualizar a posição do servo. A leitura analógica do potenciômetro retorna um valor entre 0 e 1023. Para controlar o servo nós usamos valores de 0 a 179, correspondendo ao meio giro de 180º do mesmo. Assim, é necessário usar a função **Map()** para traduzir a escala de 0-1023 para a escala de 0-179. Dessa forma, os valores lidos do potenciômetro podem ser usados como referência para determinar a posição do servomotor.

Para atualizar a posição do servo a cada iteração do loop, nós chamamos o método **write()** do objeto servomotor, passando como parâmetro o valor lido do potenciômetro traduzido para a escala de 0-179. Assim, sempre que mexermos no potenciômetro, o servo motor irá atuar e atualizar a sua posição.

Referências:

- http://blog.eletrogate.com/servo-motor-para-aplicacoes-com-arduino/
- https://www.arduino.cc/en/Tutorial/Knob
- https://www.allaboutcircuits.com/projects/servo-motor-control-with-an-arduino/
- http://www.instructables.com/id/Arduino-Servo-Motors/

Exemplo 6 - Sensor de temperatura NTC

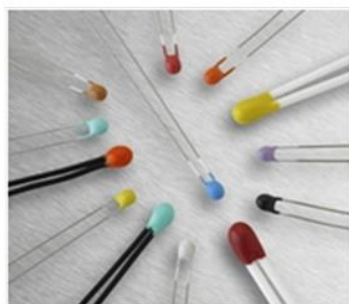
O sensor de temperatura NTC pertence a uma classe de sensores chamada de Termistores. São componentes cuja resistência é dependente da temperatura. Para cada valor de temperatura absoluta há um valor de resistência.

Assim, o princípio para medir o sinal de um termistor é o mesmo que usamos para medir o sinal do LDR. Vamos medir na verdade, a queda de tensão provocada na resistência do sensor.

Existem dois tipos básicos de termistores:

- **PTC (Positive temperature coeficient):** Nesse sensor, a resistência elétrica aumenta à medida que a temperatura aumenta;

- **NTC (Negative Temperature Coeficient):** Nesse sensor, a resistência elétrica diminui à medida que a temperatura aumenta.

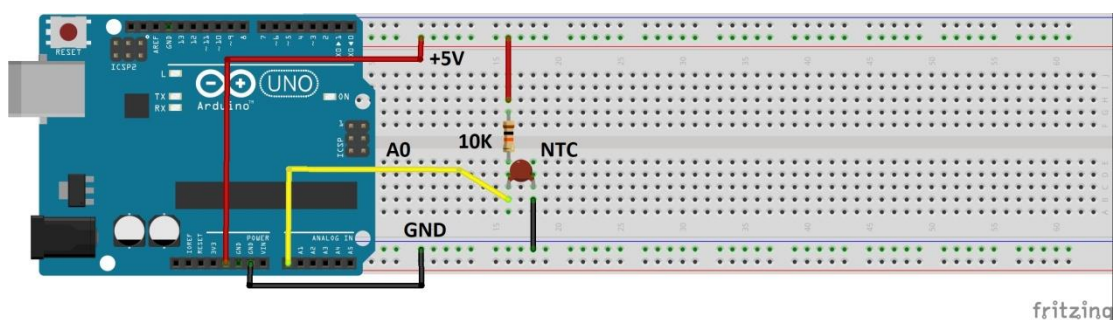
O termistor que acompanha o kit é do tipo NTC, como o próprio nome diz. Esse é o tipo mais comum do mercado.

O grande problema dos termistores é que é necessária fazer uma função de calibração, pois a relação entre temperatura e resistência elétrica não é linear. Existe uma equação, chamada equação de **Steinhart-Hart** que é usada para descrever essa relação (vide referências).

O código que vamos usar nesse exemplo utiliza a equação de Steinhart-Hart conforme a referência 1.

Lista de materiais:

Antes de mais nada, vamos separar os componentes que precisamos para montar o NTC junto com o arduino. A nossa lista de componentes é a seguinte:

- 1 Sensor de temperatura NTC;
- Arduino UNO + cabo USB;
- 1 resistor de 10k;
- Protoboard;
- Jumpers para conexão no protoboard;

Diagrama de circuito:

Para uma melhor precisão nas medidas de temperatura, meça a tensão de 5V no barramento do protoboard, usando um multímetro (não contém no KIT). Essa tensão é a mesma usada como referência do conversor analógico-digital do Arduino. Especifique esse valor de tensão na primeira linha do Sketch:

```
#define Vin 5.0
```

Por exemplo, se a tensão for 4,85 V :

#define Vin 4.85

Carregue o código abaixo e meça a temperatura com o NTC:

```
// Exemplo 6 - Sensor de temperatura NTC
// Apostila Eletrogate - KIT INTERMEDIATE

#define Vin 5.0                  // define constante igual a 5.0
#define T0 298.15                // define constante igual a 298.15 Kelvin
#define Rt 10000                 // Resistor do divisor de tensao
#define R0 10000                 // Valor da resistencia inicial do NTC
#define T1 273.15                // [K] in datasheet 0º C
#define T2 373.15                // [K] in datasheet 100° C
#define RT1 35563                // [ohms] resistencia in T1
#define RT2 549                  // [ohms] resistencia in T2
float beta = 0.0;                // parametros iniciais [K]
float Rinf = 0.0;                // parametros iniciais [ohm]
float TempKelvin = 0.0;          // variable output
float TempCelsius = 0.0;         // variable output
float Vout = 0.0;                // Vout in A0
float Rout = 0.0;                // Rout in A0

void setup()
{
  Serial.begin(9600);                     // monitor serial - velocidade 9600 Bps
  beta = (log(RT1 / RT2)) / ((1 / T1) - (1 / T2));      // calculo de beta
  Rinf = R0 * exp(-beta / T0);            // calculo de Rinf
  delay(100);                             // atraso de 100 milisegundos
}

void loop()
{
  Vout = Vin * ((float)(analogRead(0)) / 1024.0); // calculo de V0 e leitura de A0
  Rout = (Rt * Vout / (Vin - Vout));              // calculo de Rout
  TempKelvin = (beta / log(Rout / Rinf));         // calculo da temp. em Kelvins
  TempCelsius = TempKelvin - 273.15;              // calculo da temp. em Celsius
  Serial.print("Temperatura em Celsius: ");       // imprime no monitor serial
  Serial.print(TempCelsius);                      // imprime temperatura Celsius
  Serial.print("    Temperatura em Kelvin: ");    // imprime no monitor serial
  Serial.println(TempKelvin);                     // imprime temperatura Kelvins
  delay(500);                                     // atraso de 500 milisegundos
}
```

Referências e sugestões de leitura:

1. http://www.instructables.com/id/NTC-Temperature-Sensor-With-Arduino/
2. https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation
3. http://www.thinksrs.com/downloads/PDFs/ApplicationNotes/LDC%20Note%204%20NTC%20Calculator.pdf
4. https://www.mundodaeletrica.com.br/sensor-de-temperatura-ntc-ptc/

Exemplo 7 - Módulo Relé 1 Canal

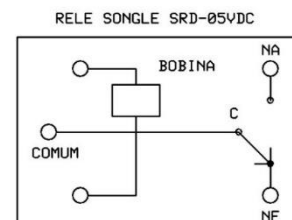
O Módulo Relé nada mais é do que um ou mais relés acionados por sinais digitais de 5V. Esse componente é indicado para acionar cargas que utilizam correntes maiores do que a suportada pelo Arduino, ou então uma carga de um circuito de corrente alternada (rede elétrica).

O módulo de relé pode ser usado para vários tipos de aplicações:

- ativação de eletroímãs,
- ativação de motores CC,
- ativação de motores CA,
- ligar/desligar lâmpadas.

O terminal de controle do relé (bobina) é ligado a um dos pinos digitais do Arduino. O outro terminal da carga é ligado ao outro terminal da tomada (ou ao outro terminal da bateria).

Uma das aplicações mais comuns dos módulos relés para arduino é no acendimento de lâmpadas em sistemas de automação residencial. O módulo funciona exatamente da mesma forma que uma chave ou interruptor. Esse é o diagrama de um relé SONGLE SRD-05VDC-SL-C:

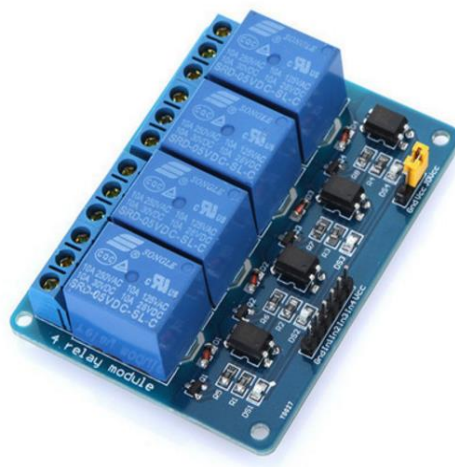
- NA - Contato Normalmente Aberto
- C - Terminal Comum
- NF - Contato Normalmente Fechado

O terminal C é o Comum do Contato. Os outros dois terminais usamos para ligar ou desligar a carga. Se ligamos a carga no terminal NA, a carga será acionada setando o pino de controle do relé para nível alto. Se ligarmos a carga no pino NF, ela será ativada

setando o pino de controle do Arduino para nível baixo, e desligada setando o pino de controle para nível alto.

A capacidade corrente em cada contato é de 10 A. Não ultrapasse esse valor, para não danificar o seu relé.

Os outros dois terminais são para alimentação da bobina do relé – no caso 5V . O terminal de controle do Módulo esta conectado nessa bobina. O consumo de corrente da bobina do relé é de aproximadamente 90 mA, quando energizada.

No mercado você pode encontrar módulos de 1 , 2, 4 e 8 canais. Cada canal corresponde a um relé montado no módulo. Ou seja, um módulo relé de 5V e 8 canais, é um conjunto de 8 relés acionados por sinais separados de 5V. Um módulo relé de 12V e 4 canais, nada mais do que um conjunto de 4 relés, cada qual acionado por sinais de 12V.

Na imagem abaixo temos como exemplo um relé de 4 canais.

Módulo relé de 4 canais. Créditos: Eletrogate

Repare que cada relé possui os três terminais que explicamos mais acima. Além deles, há outros 6 pinos usados para interfacear o módulo com o Arduino. São eles:

- GND,
- IN1,
- IN2,
- IN3,

- IN4,
- VCC.

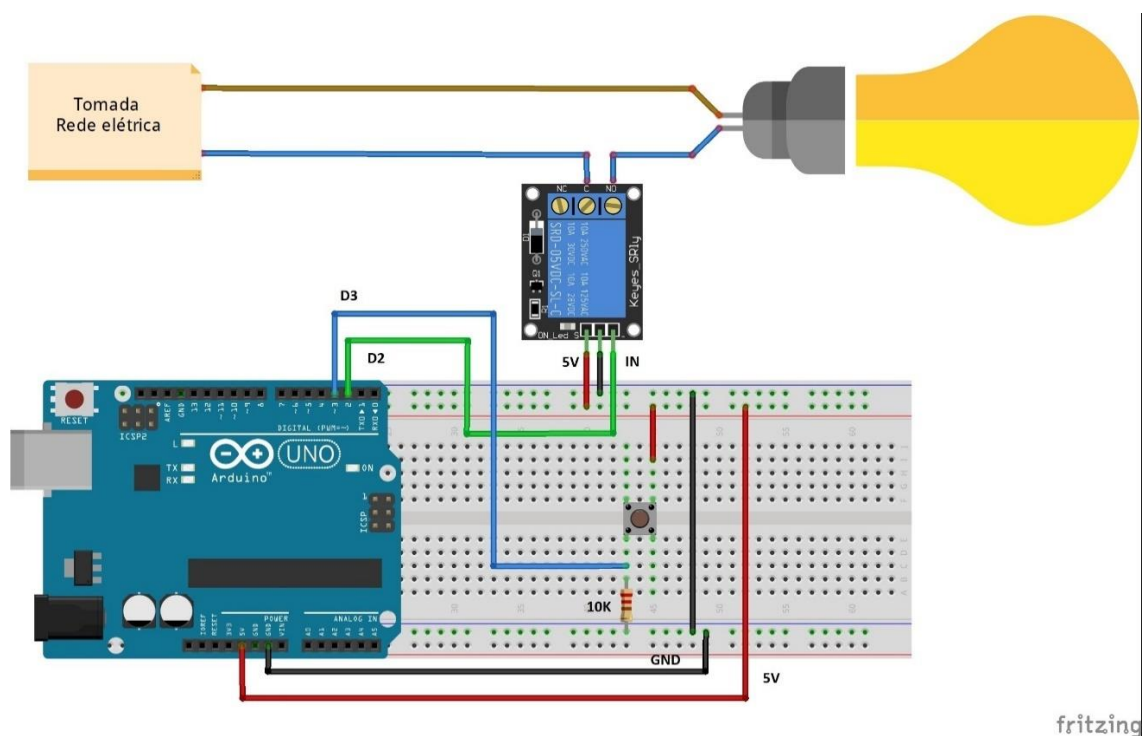
Cada pino **INX** serve para acionar um dos relés. **VCC** e **GND** são os pinos de alimentação do Módulo do relé : VCC deve ser conectado no +5V e o GND no terra, ambos da fonte. **O terra (GND) do Módulo Relé deve estar conectado também no Terra (GND) do Arduino !**

Lista de Materiais:

Para este exemplo você vai precisar:

- Protoboard,
- Arduino UNO,
- Jumpers de ligação,
- Módulo relé de 1 canal,
- Uma chave táctil (push-button),
- 1 resistor de 10K.

ATENÇÃO = se você não tem conhecimentos em montagens de circuitos elétricos CA, procure uma pessoa especializada para montar para você. Risco de choque elétrico ! Verifique todas as ligações, antes de energizar o circuito ! A Eletrogate não se responsabilizará por danos materiais ou pessoais no caso de acidentes.

Diagrama de circuito:

Código para acionar o Relé com uma chave táctil:

```
// Exemplo 7 - Módulo Relé 1 Canal
// Apostila Eletrogate - KIT BEGINNING

int buttonPin = 3;                        // pino D3 para botao
int ledPin =  13;                         // LED na placa Arduino
int IN1 = 2;                              // pino D2 para controle do rele

int ledState = LOW;                       // estado do LED = LOW
int buttonState;                          // variavel do estado do botao
int lastButtonState = LOW;                // estado previo do botao = LOW

unsigned long lastDebounceTime = 0;       // ultimo tempo de debounce
unsigned long debounceDelay = 50;         // timer do debounce 50 ms

void setup()
{
  pinMode(ledPin, OUTPUT);                // pino do LED = saida
  pinMode(buttonPin, INPUT);              // pino do botao = entrada
  pinMode(IN1, OUTPUT);                   // pino de controle do rele = saida
  digitalWrite(ledPin, ledState);         // apaga o LED
  digitalWrite(IN1, ledState);            // desliga o rele
}

//CONTINUA NA PRÓXIMA PÁGINA!
```

```
 void loop()
{
  int reading = digitalRead(buttonPin);    // le o estado do botao
  if (reading != lastButtonState)          // se o estado for diferente do anterior
  {
    lastDebounceTime = millis();           // reset do timer debouncing
  }

  if ((millis() - lastDebounceTime) > debounceDelay)
  {
    // se o tempo do botao pressionado for maior do que debounce
    if (reading != buttonState)            // se o estado do botao for diferente do anterior
    {
      buttonState = reading;               // muda o estado do botao
      if (buttonState == HIGH)             // se o botao esta no nivel High
      {
        ledState = !ledState;              // muda o estado do LED
      }
    }
  }
  digitalWrite(ledPin, ledState);          // seta o LED
  digitalWrite(IN1, ledState);             // seta o estado do rele 1
  lastButtonState = reading;               // salva a leitura do botao
}
```

Referências:

- http://blog.eletrogate.com/modulo-rele-para-automacao-residencial-com-arduino/
- https://www.allaboutcircuits.com/projects/use-relays-to-control-high-voltage-circuitswwith-an-arduino/
- https://www.arduino.cc/en/Tutorial/Debounce

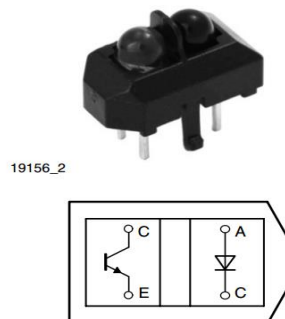
Exemplo 8 - Sensor Ótico Reflexivo TCRT5000

O sensor ótico reflexivo **TCRT5000** é um dos mais populares para utilização em projetos com Arduino. O sensor é fabricado pela Vishay, uma tradicional fabricante de componentes eletrônicos. Recomendamos fortemente, a leitura do datasheet do sensor:

https://www.vishay.com/docs/83760/tcrt5000.pdf

Trata-se de um sensor reflexivo que possui um emissor infravermelho e um fototransistor. O emissor é um led infravermelho que emite um sinal nessa faixa do espectro. Já o fototransistor é o receptor que faz a leitura do sinal refletido. Ou seja, o led emite um feixe infravermelho que pode ou não ser refletido por um objeto. Caso o feixe seja refletido, o fototransistor identifica o sinal refletido e gera um pulso em sua saída.

Tensão direta do LED emissor é de 1,25 V com uma corrente máxima de 60 mA. A corrente máxima do fototransistor é de 100 mA.

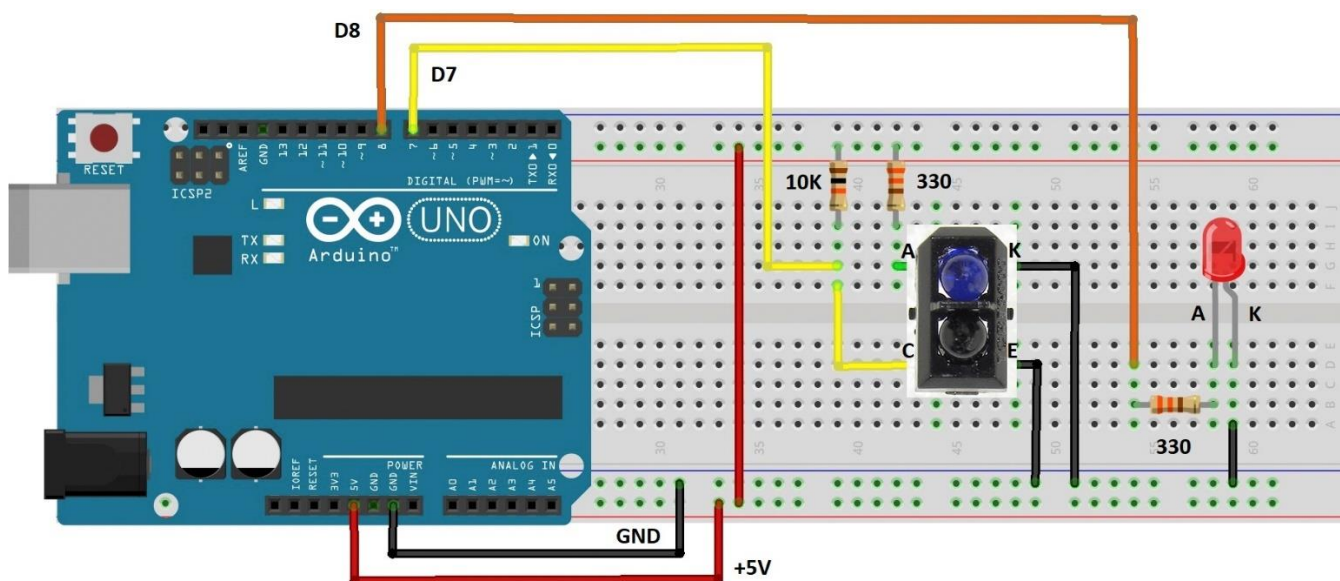
A distância máxima de detecção não é grande, ficando em torno de 25 mm (dois centímetros e meio), o que pode limitar um pouco a sua aplicação. O fototransistor vem com um filtro de luz ambiente, o que maximiza a identificação do feixe infravermelho refletido.

Sensor TCRT5000 visto por cima   Fonte: Vishay

Lista de materiais:

- Arduino UNO R3;
- Protoboard;
- Sensor TCRT5000;
- Led vermelho 5mm;
- Resistor de 330;
- Resistor de 10K;
- Jumpers para ligação no protoboard;

Diagrama de Circuito:

O terminal Anodo (A) positivo do LED infravermelho está ligado por meio de um resistor de 330R ao VCC (5V) e o seu terminal catodo (K) negativo está ligado em GND. O Coletor (C) do fototransistor está ligado ao resistor de 10K (que esta conectado no VCC) e também na entrada digital D7 do Arduino. Essa entrada é que vamos usar para identificar se o sensor percebeu algum objeto ou não. O Emissor (E) do fototransistor está ligado ao GND.

O LED vermelho para indicação do Sensor, tem o terminal Anodo (A) positivo conectado à um resistor de 330R que está conectado à porta digital D8 do Arduino. O terminal catodo (K) negativo está ligado ao GND.

Código:

```
// Exemplo 8  - Sensor Ótico Reflexivo TCRT5000
// Apostila Eletrogate - KIT INTERMEDIATE

int leituraSensor;                    // variavel de leitura do sensor
int led = 8;                          // led indicador = D8 do Arduino
int fotoTransistor = 7;               // coletor do fototransistor = D7 do Arduino

void setup()
{
  pinMode(led, OUTPUT);               // pino do led indicador = saida
  pinMode(fotoTransistor, INPUT);     // pino do coletor do fototransistor = entrada
}

void loop()
{
  leituraSensor = digitalRead(fotoTransistor);  // leitura do sensor TCRT5000
  if (leituraSensor == 0 )                      // se o led refletir a luz
    digitalWrite(led, HIGH);                    // acende LED indicador
  else                                          // senão
    digitalWrite(led, LOW);                     // apaga LED indicador
  delay(500);                                   // atraso de 0,5 segundos
}
```

O código para utilização do sensor TCRT5000 é bem simples. Com o circuito montado, basta monitorar o estado do pino no qual a saída do sensor (coletor do fototransistor) está ligada. Se esse pino estiver em nível baixo, é porque algum objeto está presente dentro do raio de medição do sensor (a luz infra-vermelha do LED esta sendo refletida). Se o pino estiver em nível alto, então não há objeto nenhum no raio de medição do sensor.

Nesse exemplo, a presença do sensor é identificada pelo acionamento de um led, ou seja, sempre que um objeto for identificado pelo sensor, o led ficará aceso.

Referências:

- http://blog.eletrogate.com/sensor-optico-tcrt5000-com-arduino/
- http://www.instructables.com/id/Using-IR-Sensor-TCRT-5000-With-Arduino-and-Program

Exemplo 9 - Display LCD 16x2

O display 16×2 é um dispositivo que possui interface para comunicação e controle padronizada. Esses displays são fáceis de serem controlados e graças à essa padronização é possível trocar um display 16×2 de um fabricante por outro diferente sem maiores problemas.

A tecnologia utilizada nos displays tradicionais é LCD (Liquid Crystal Display). Mais recentemente, modelos mais modernos com tecnologia O-Leds , os LEDS orgânicos, já estão sendo encontrados também.

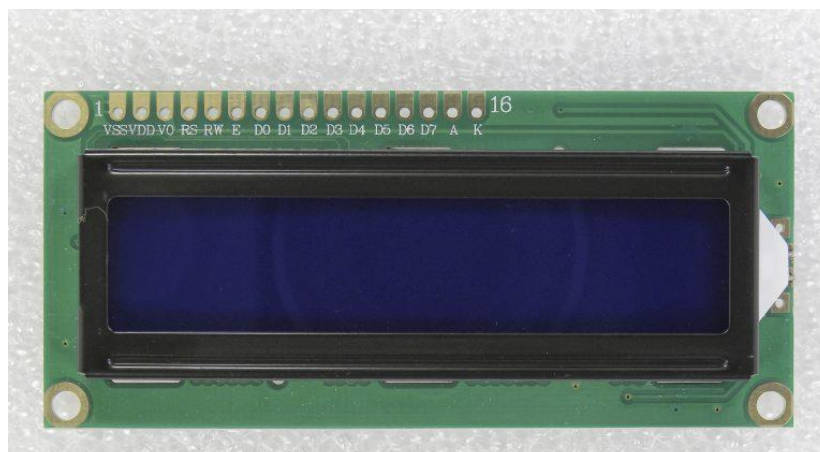
A grande vantagem e razão pela qual até hoje é componente popular, é o seu preço e a facilidade de implementação em projetos eletrônicos. Antes dos displays LCD, a interface gráfica usada para mostrar caracteres alfa numéricos era os displays a LED de x segmentos (os mais comuns eram de 7, 14 ou 16 segmentos).

Esses componentes mais antigos utilizavam vários leds (7,14 ou 16) dispostos em segmentos, de forma que dependendo de qual conjunto de leds fosse aceso, um determinado caractere seria mostrado.

LCD 16x2. Créditos: Eletrogate

Na hora de comprar um display LCD, você vai encontrar diversas opções, além do mais tradicional 16×2. A especificação é dada em relação ao número de caracteres por linha e o número de linhas. Assim, 16×2 significa que o display pode exibir 16 caracteres em cada linha, e possui 2 linhas. Alguns valores típicos para essas especificações são:

- Quantidade de caracteres padrão de mercado: 8, 12, 16, 20, 24 e 40;
- Quantidade de linhas mais comuns: 1, 2 e 4;
- Cores de fundo (backlight): Azul ou verde;

O display que vamos trabalhar é de 16 colunas e 2 linhas (16×2) e com backlight azul e caracteres brancos. A interface com Arduino é feita por meio de 16 pinos. Em geral apenas 12 são utilizados de fato. Os pinos do LCD são:

LCD 16x2 pinagem – foto Gustavo Murta

- Pino de **register select (RS):** Controle em que parte da memória (do LCD) o usuário está escrevendo os dados. São duas opções, você pode escrever um dado para ser mostrado no display, ou uma instrução no MCU do display;
- Pino de **Read/Write (R/W)**: Seleciona se está em modo de leitura ou de escrita;
- Pino de **Enable**: Habilita a escrita de dados nos registradores;
- 8 pinos de dados **(D0 -D7).** Os estados de cada pino desses corresponde aos bits que você está escrevendo em um registrador do MCU (do display), ou os valores que você lê de um registrador quando está no modo de leitura.
- Pinos de alimentação **VCC e GND** do LCD e do LED Backlight (A-anodo/K-catodo);
- Pino **V0** de ajuste de contraste;

Vamos conferir em nosso exemplo como usar o LCD 16x2. Na figura abaixo mostramos um outro modelo, um display gráfico de 128 colunas e 64 linhas, também muito comercializado e que você pode obter separadamente do Kit para implementar mensagens gráficas mais avançadas.

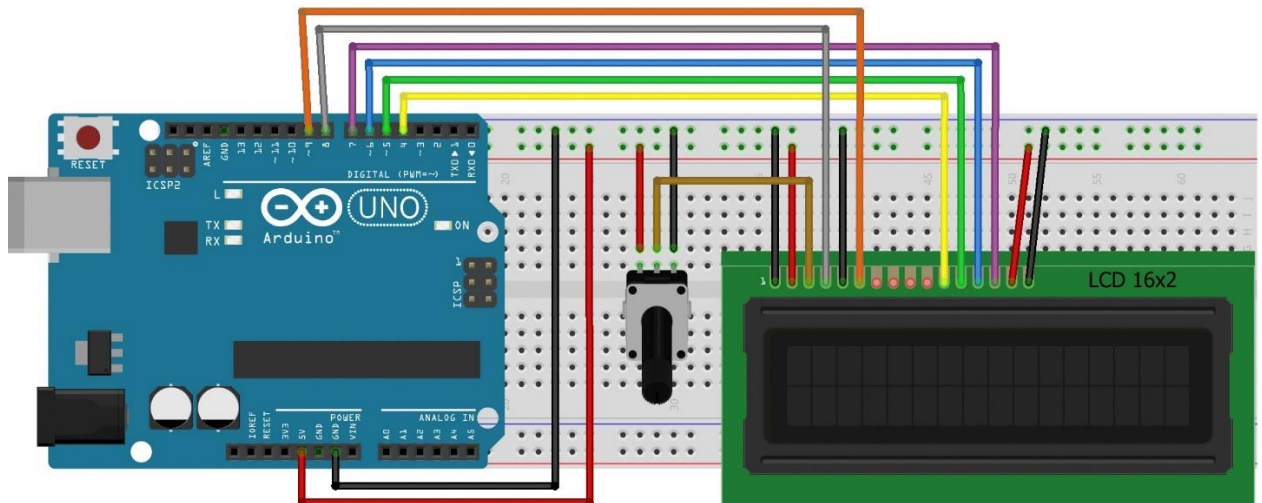
Display gráfico 128x64

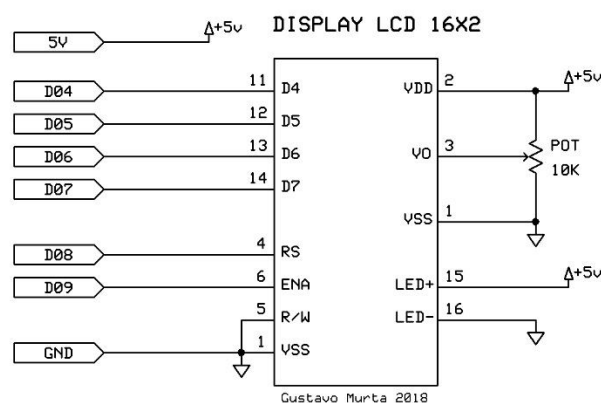
Lista de materiais:

A lista de materiais é a seguinte:

- 1 Arduino UNO;
- 1 Protoboard;

- Jumpers para ligação no protoboard;
- Display LCD 16x2;
- Potenciômetro 10K;

Diagrama de circuito:

No diagrama acima, o potenciômetro é usado para controlar o contraste do LCD, e deve ser ajustado sempre que você ligar o display para poder visualizar bem cada caracter. As demais ligações são padrão. No código da próxima seção, os pinos do Arduino são configurados de acordo com a ligação acima.

Veja o diagrama eletrônico da montagem acima:

Código:

```
// Exemplo 9 - Display LCD 16x2
// Apostila Eletrogate - KIT INTERMEDIATE

#include <LiquidCrystal.h>              // biblioteca Liquid Crystal

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);    // pinos do LCD - RS E D4 D5 D6 D7
int contador = 0;                       // variável contador

void setup()
{
  lcd.begin(16, 2);                     // inicializa LCD 16x2
  delay(500);                           // atraso de 0,5 segundos
}

void loop()
{
  lcd.clear();                          // limpa tela do LCD
  lcd.setCursor(0, 0);                  // selecionando coluna 0 e linha 0
  lcd.print("Exemplo LCD !");           // mostra no LCD
  lcd.setCursor(1, 1);                  // selecionando coluna 1 e linha 1
  lcd.print(contador);                  // mostra no LCD a contagem
  contador++;                           // incrementa contador
  if (contador == 60)                   // se contador = 60
    contador = 0;                       // zera o contador
  delay(1000);                          // atraso de 1 segundo
}
```

No código, é mostrado na primeira linha (0) do display, a mensagem “Exemplo LCD !“. Na segunda linha (1) será mostrado um contador que é incrementado de 1 em 1 segundo. Para usar o LCD é preciso incluir a biblioteca **LiquidCrystal.h**, que é nativa da IDE arduino. Essa biblioteca possui as seguintes funções:

- lcd.clear() - Limpa a tela do display

- lcd.SetCursor(0,0) - Posiciona o cursor a partir de onde os caracteres serão escritos
- lcd.print() - Imprime caracteres no display

No exemplo são usadas essas três funções, mas a biblioteca dispõe de várias outras que você pode conferir na página oficial (vejas as referências abaixo).

Referências:

- http://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/
- http://blog.eletrogate.com/display-lcd-modulo-rf-para-deteccao-de-presenca/
- https://www.arduino.cc/en/Reference/LiquidCrystal

Exemplo 10 - Sensor de Distância Ultrassônico HC-SR04

O princípio de funcionamento do Sensor HC-SR04 consiste na emissão de sinais ultrassônicos e na leitura do sinal de retorno (reflexo/eco) desse mesmo sinal. A distância entre o sensor e o objeto que refletiu o sinal é calculada com base no tempo entre o envio e leitura de retorno.

"Sinais Ultrassônicos são ondas mecânicas com frequência acima de 40 KHz"

Como ouvido humano só consegue identificar ondas mecânicas até a frequência de 20KHz, os sinais emitidos pelo sensor Ultrassônico não podem ser escutados por nós.

O sensor HC-SR04 é composto de três partes principais:

- Transmissor Ultrassônico – Emite as ondas ultrassônicas que serão refletidas pelos obstáculos;
- Um receptor – Identifica o eco do sinal emitido pelo transmissor;
- Circuito de controle – Controla o conjunto transmissor/receptor, calcula o tempo entre a emissão e recepção do sinal;

O sensor é ilustrado na imagem abaixo. Repare como os invólucros do transmissor e receptor são bem proeminentes.

Sensor HC-SR04

O funcionamento consiste em três etapas:

1. O circuito externo (Arduino) envia um pulso de trigger de pelo menos 10us em nível alto;
2. Ao receber o sinal de trigger, o sensor envía 8 pulsos de 40khz e detecta se há algum sinal de retorno ou não;
3. Se algum sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico;

Por meio desse pulso de saída cujo tempo é igual a duração entre emissão e recepção, nós calculamos a distância entre o sensor e o objeto/obstáculo, por meio da seguinte equação:

$$Distancia = \frac{(Tempodeduraçãodosinaldesaídaxvelocidadedosom)}{2}$$

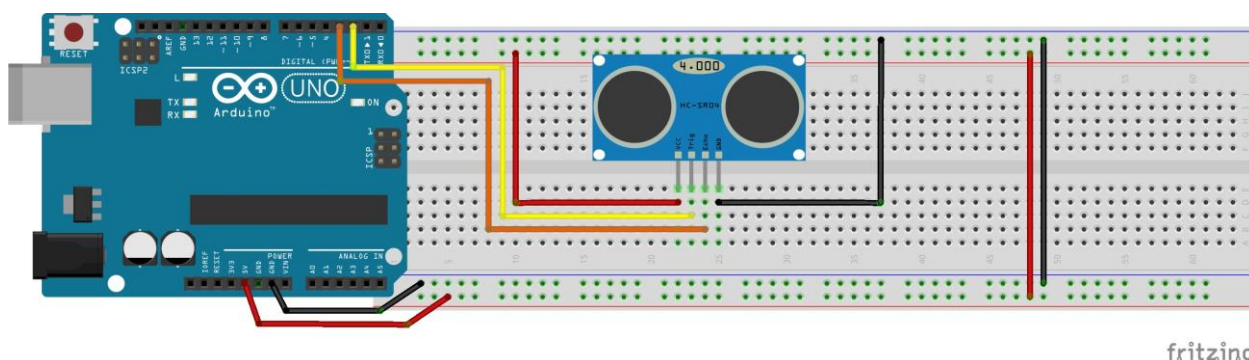
Em que o Tempo de duração do sinal de saída é o tempo no qual o sinal de output permanece em nível alto e a velocidade do som = 340 m/s;

Repare que as unidades devem estar coerentes para o resultado da conta ser correto. Assim, se a velocidade do som é dada em metros/segundo, o tempo de duração do sinal de saída deve estar em segundos para que possamos encontrar a distância em metros.

Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Protoboard;
- Jumpers de ligação;
- Sensor ultrassônico HCSR-04

Diagrama de circuito:

A montagem é bem direta, basta interligar o sensor com o Arduino. Não há necessidade de conversão de níveis lógicos ou ligação de componentes externos.

Código :

As variáveis declaradas são para determinar os pinos de trigger (pino 2) e de leitura do sensor (pino 3). Temos três variáveis do tipo float utilizadas para medir o tempo do pulso no output do sensor e calcular a distância. Na função void setup(), inicializamos o pino 2 como saída e o 3 com entrada. Além disso, configuramos a comunicação serial para 9600 de baud rate.

Na função void loop(), onde o programa será executado continuamente, executa-se três passos básicos:

1. Enviar pulso de 10us para o pino de trigger do sensor. Isto é feito com a função DisparaPulsoUltrassonico();

2. Ler o pino de output do sensor e medir o tempo de duração do pulso utilizando a função PulseIn. Esta função retorna o tempo de duração do pulso em microsegundos para que o mesmo seja armazenado em uma variável;

3. Por fim, chamamos a função CalculaDistancia (tempo) passando como parâmetro o tempo lido com a função PulseIn. Esta função calcula a distância entre o sensor e o obstáculo. Nós chamamos esta função de dentro da função

```
// Exemplo 10 - Sensor de Distância Ultrassônico HC-SR04
// Apostila Eletrogate - KIT BEGINNING

int PinTrigger = 2;                          // pino usado para disparar os pulsos do
sensor
int PinEcho = 3;                             // pino usado para ler a saida do sensor
float TempoEcho = 0;                         // variável tempo do eco
const float velocidadeSom_mps = 340;         // em metros por segundo
const float velocidadeSom_mpus = 0.000340;   // em metros por microsegundo

void setup()
{
  pinMode(PinTrigger, OUTPUT);               // configura pino Trigger como saída
  digitalWrite(PinTrigger, LOW);             // pino trigger - nível baixo
  pinMode(PinEcho, INPUT);                   // configura pino ECHO como entrada
  Serial.begin(9600);                        // inicializa monitor serial 9600 Bps
  delay(100);                                // atraso de 100 milisegundos
}

void loop()
{
  DisparaPulsoUltrassonico();                     // dispara pulso ultrassonico
  TempoEcho = pulseIn(PinEcho, HIGH);             // mede duração do pulso HIGH de eco em
microsegundos
  Serial.print("Distancia em metros: ");          // mostra no monitor serial

  Serial.println(CalculaDistancia(TempoEcho));    // mostra o calculo de distancia em
metros
  Serial.print("Distancia em centimentros: ");    // mostra no monitor serial
  Serial.println(CalculaDistancia(TempoEcho) * 100); // mostra o calculo de distancia em cm
  delay(2000);                                    // atraso de 2 segundos
}
//CONTINUA NA PRÓXIMA PÁGINA
```

Serial.println(), de forma que o resultado já é impresso diretamente no terminal serial;

```
void DisparaPulsoUltrassonico()
{
  digitalWrite(PinTrigger, HIGH);                    // pulso alto de Trigger
  delayMicroseconds(10);                             // atraso de 10 milisegundos
  digitalWrite(PinTrigger, LOW);                     // pulso baixo de Trigger
}

float CalculaDistancia(float tempo_us)
{
  return ((tempo_us * velocidadeSom_mpus) / 2 );     // calcula distancia em metros
}
```

A função **DisparaPulsoUltrassonico()** apenas ativa o pino 2 em nível alto, espera 10 microsegundos e volta a setar o pino para nível baixo. Lembre que 10 us é o tempo mínimo que o pulso deve perdurar para disparar o sensor HC-SR04.

A função **CalculaDistancia()** recebe como parâmetro o tempo do pulso no pino Echo do sensor. Com esse tempo nós usamos a equação, para calcular a distância entre o sensor e o obstáculo.

Referências:

- https://create.arduino.cc/projecthub/dusnoki/arduino-ultrasonic-sensor-hc-sr04-full-build-and-code-73614d
- http://blog.eletrogate.com/sensor-ultrassonico-hc-sr04-com-arduino/

Exemplo 11 – Motor de passo unipolar 28BYJ-48 + Driver ULN2003

O Motor de passo é um motor elétrico que não possui escovas ou comutadores, permitindo assim uma vida longa sem tantos desgastes. O rotor (parte móvel) constitui-se de um ou mais imãs permanentes. No estator (parte fixa) encontram-se duas ou mais bobinas, dependendo do tipo de motor. O controle do motor é feito por um circuito eletrônico que aciona repetidamente as bobinas numa sequência que permite o giro do rotor. Cada pulso enviado para o circuito, faz com que o motor avance um passo. O sentido de rotação do motor é controlado também pela sequência e pela polarização das bobinas. A velocidade que o rotor gira é determinada pela frequência dos pulsos do circuito de controle. Quanto maior a frequência, maior será o RPM.

Informações sobre o Motor de Passo 28BYJ-48:

Esse motor é Unipolar pois possui 4 enrolamentos que chamamos de fases. Em uma das pontas das fases, todas estão conectadas juntas (fio comum). Portanto, esse motor não pode ser usado em Drivers para motores Bipolares (com duas fases somente). Interessante, que ele pode ser alimentado com 5V e consome baixa corrente, o que facilita a montagem.

Algumas características do motor:

- Tensão de operação : 5V CC
- Número de fases : 4
- Razão da variação de velocidade : 1/64 (mecanismo de redução)
- Angulo do passo : 5,625 graus => 64 passos/volta (360/64=5,625)
- Resistência CC : 23 ohms
- Frequência : 100 Hz
- Torque de tração: > 34,3 mN.m

OBS: a resistência medida nas bobinas do motor foi de 23 ohms. Assim podemos deduzir o consumo estático de corrente em cada bobina :

I = V / R = 5V / 23 ohms = 217 mA

Nesse tipo de motor, cada fase é energizada por um circuito driver num único sentido de corrente. Isto é, uma extremidade do enrolamento será sempre positiva e a outra sempre negativa. A desvantagem do motor unipolar é que tem menos torque do que o bipolar similar, pois sempre terá no máximo, a metade das fases energizadas. Daí pode-se concluir que tem 50% da eficiência em relação ao bipolar.

Esse motor tem uma caixa de engrenagens para redução da rotação do eixo, mas que permite um aumento no torque. Veja abaixo a caixa de engrenagens desmontada. A relação de redução é de 63,68 , isto é para cada giro do eixo reduzido, são necessárias 63,68 voltas do motor de passo interno. Como o motor de passo (interno) precisa de 64 passos para dar uma volta, são necessários 64 x 63,68 = 4075 passos aproximadamente para um giro do eixo reduzido.

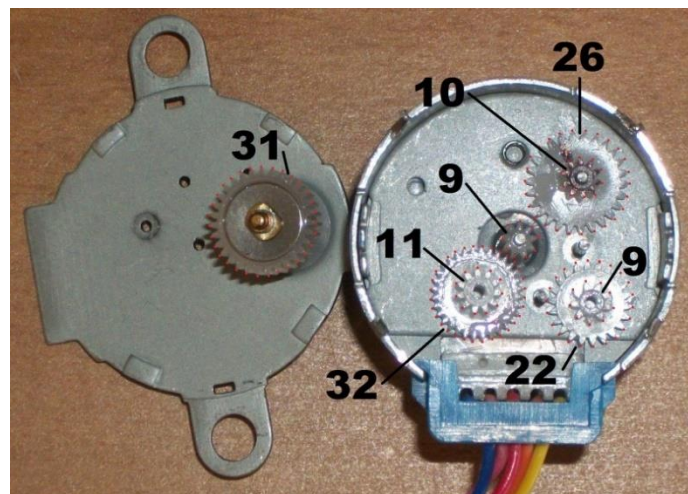
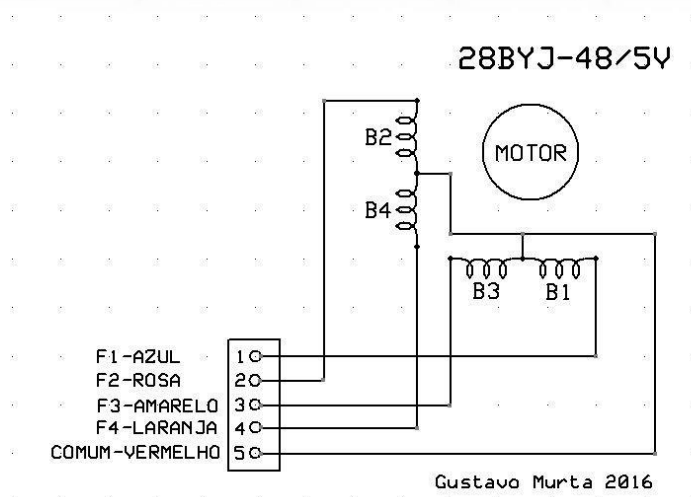
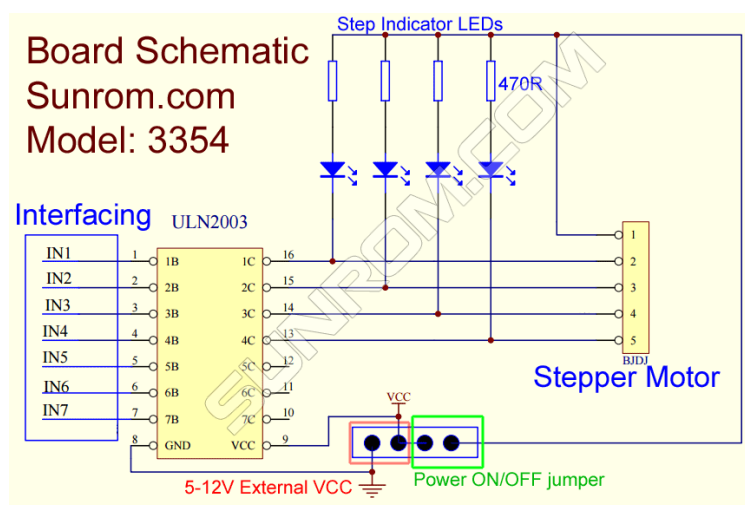
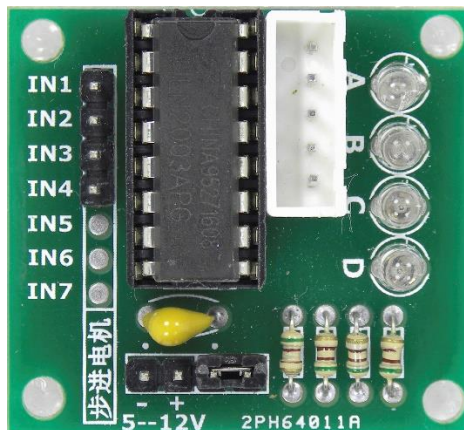



Diagrama do motor 28BYJ-48:

Informações sobre o Módulo Driver ULN2003:

O único chip no módulo é o ULN2003. Esse chip possui um conjunto de sete drivers de transistores Darlington que permitem o acionamento de cargas indutivas. Todas as saídas tem o coletor aberto e diodos de supressão (Clamp) . Os transistores suportam tensões de até 50V e correntes de até 500 mA. Todas as entradas IN1, IN2,IN3 e IN4 são compatíveis com sinais TTL e CMOS, com limite de 5V. O pino comum tem que ser conectado na tensão de alimentação do motor. Nesse caso é conectado no 5V.

Esse é o diagrama da placa de circuito do Módulo ULN2003APG :

Pinos do Módulo ULN2003APG :

O pino mais à esquerda (-) no módulo é o terra (veja foto acima). Esse terra tem que ser conectado ao terra da fonte e ao terra do Arduino. O pino (+) no caso do Motor 28BYJ-48, tem que ser conectado ao positivo de uma fonte de 5V (preferencialmente de 1 Ampére). Não recomendo que conecte no 5V do Arduino, pois poderá sobrecarregar o regulador de tensão do mesmo. O jumper Power ON/OFF é usado para ligar ou desligar o motor.

Os quatro leds vermelhos (A,B,C e D) são usados para indicar o acionamento de cada um dos drivers (fases do motor). Mesmo que o diagrama mostre sete entradas, somente quatro podem ser usadas e somente essas tem pinos no conector (IN1,IN2,IN3 e IN4).

Muita atenção ao conectar a alimentação nos pinos. Uma ligação incorreta poderá danificar o driver ou o motor.

Motor de Passo – modos de operação:

Para um motor de Passo Unipolar, temos alguns modos de operação. Temos o **modo Passo completo com alto torque** (Full step), o **modo Passo completo com baixo torque** (Wave Step), o **modo Meio Passo** (half step) e **Micro-passo** (Micro stepping).

No caso do Micro passo, a corrente nos enrolamentos deve ser alterada. Como o nosso circuito não permite o controle da corrente, esse modo de operação não se aplica para esse driver.

Passo completo com alto torque (Full step): - Duas Fases são acionadas ao mesmo tempo.

Step	ϕ4	ϕ3	ϕ2	ϕ1
0	0	0	1	1
1	0	1	1	0
2	1	1	0	0
3	1	0	0	1

Passo completo com baixo torque (Wave Step): - Somente uma Fase acionada de cada vez.

Step	ϕ4	ϕ3	ϕ2	ϕ1
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

Meio Passo (half step): - Na sequência de oito passos, em alguns passos temos somente uma fase acionada e em outros passos, temos duas fases acionadas.

Step	ϕ4	ϕ3	ϕ2	ϕ1
0	0	0	0	1
1	0	0	1	1
2	0	0	1	0
3	0	1	1	0
4	0	1	0	0
5	1	1	0	0
6	1	0	0	0
7	1	0	0	1

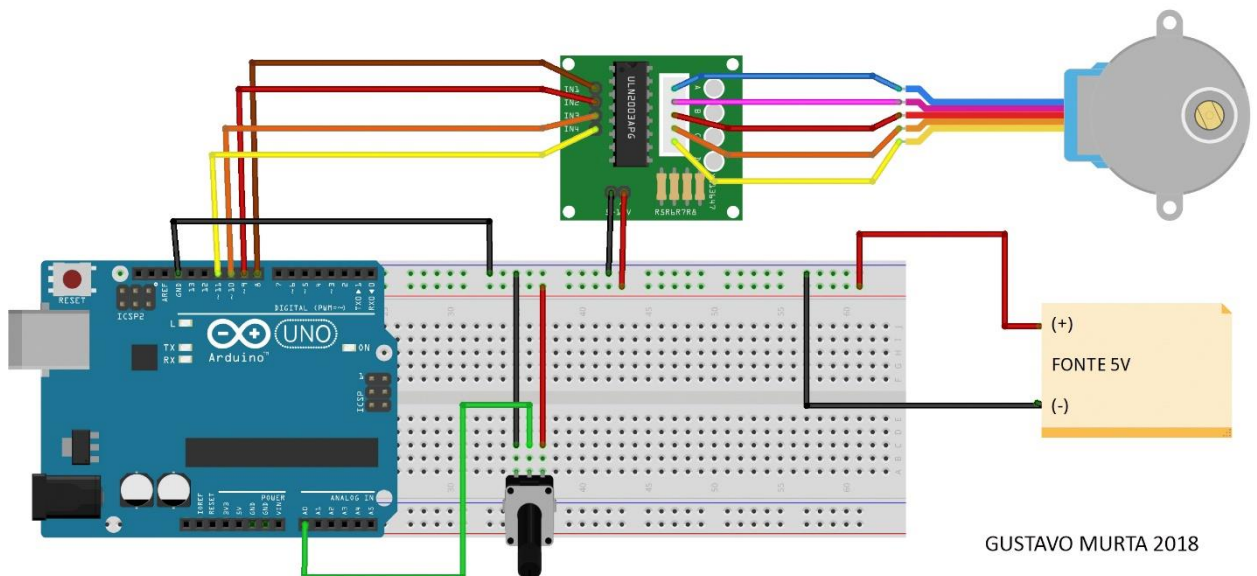
Montagem do Motor e do Driver com Arduino:

Esse motor de passo pode ser usado em inúmeras aplicações. Nesse caso, como a rotação é baixa, pode ser usado em uma mini Plotter, uma mini CNC Laser, controle de movimento de câmera fotográfica e muito mais.

Nessa montagem do Kit , o Skecth usará a Biblioteca **AccelStepper.** Para instalar a nova Biblioteca, clique em **Sketch > Incluir Biblioteca > Gerenciar Bibliotecas** .

Após abrir a janela do **Gerenciador de Biblioteca**, refine a busca digitando o nome da biblioteca. Na biblioteca selecionada, clique em **More Info** e depois em Instalar. Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet. Após a instalação da biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

Nessa aplicação, gire o potenciômetro e o motor irá acompanhar o movimento do mesmo (giro total de 180 graus do eixo do motor). Use um potenciômetro de 10 K ohms. Recomendo que use uma fonte externa de 5V (não incluída no KIT) para alimentar o driver e o motor.

Diagrama do circuito :

Esse é o Sketch da aplicação:

```
// Exemplo 11 - Motor de passo unipolar 28BYJ-48 + Driver ULN2003

// Apostila Eletrogate - KIT INTERMEDIATE

// Baseado em AccelStepper/ProportionalControl 8pde-example



#include <AccelStepper.h>          // biblioteca AccelStepper

#define ANALOG_PIN A0              // pino A0 para leitura da tensão do Potenciometro

AccelStepper motorPasso (AccelStepper::FULL4WIRE, 8, 10, 9, 11);  // Passo completo

void setup()
{
  motorPasso.setMaxSpeed(500);              // maxima velocidade = 500 pulsos/seg
}

void loop()
{
  int analog_in = analogRead(ANALOG_PIN);   // lendo a tensão do pino A0 do Arduino
  motorPasso.moveTo(analog_in);             // gira o eixo do motor X pulsos
  motorPasso.setSpeed(100);                 // velocidade = 100 pulsos por segundo
  motorPasso.runSpeedToPosition();          // gira o eixo para a posição definida
}
```

Referências :

- http://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003/
- http://blog.eletrogate.com/driver-a4988-motor-de-passo-usando-o-arduino/

Considerações finais

Essa apostila tem por objetivo apresentar alguns exemplos básicos sobre como utilizar os componentes do Kit Arduino INTERMEDIATE, a partir dos quais você pode combinar e fazer projetos mais elaborados por sua própria conta.

Nas seções de referências de cada exemplo e nas referências finais, também tentamos indicar boas fontes de conteúdo objetivo e com projetos interessantes. Sobre isso ponto, que consideramos fundamental, gostaríamos de destacar algumas fontes de conhecimento que se destacam por sua qualidade.

O fórum oficial Arduino possui muitas discussões e exemplos muito bons. A comunidade de desenvolvedores é bastante ativa e certamente pode te ajudar em seus projetos. No Project Hub poderá encontrar milhares de projetos com Arduino.

- Fórum oficial Arduino: https://forum.arduino.cc/
- Project Hub Arduino : https://create.arduino.cc/projecthub

O Instructables é a ótima referência do mundo maker atual. Pessoas que buscam construir suas próprias coisas e projetos encontram referências e compartilham suas experiências no site.

- Instructables: https://www.instructables.com/

O Maker pro é outro site referência no mundo em relação aos projetos com Arduino. Há uma infinidade de projetos, todos bem explicados e com bom conteúdo.

- Maker pro: https://maker.pro/projects/arduino

Em relação à eletrônica, teoria de circuitos e componentes eletrônicos em geral, deixamos alguns livros essenciais na seção finais de referências. O leitor que quer se aprofundar no mundo da eletrônica certamente precisará de um livro basilar e de bons conhecimentos em circuitos eletro-eletrônicos.
No mais, esperamos que essa apostila seja apenas o início de vários outros projetos e, quem sabe, a adoção de Kits mais avançados, como os de **Robótica** e **Arduino Advanced**.

Qualquer dúvida, sugestão, correção ou crítica a esse material, fique à vontade para relatar em nosso blog oficial: http://blog.eletrogate.com/

Referências gerais

1. Fundamentos de Circuitos Elétricos. Charles K. Alexander; Matthew N. O. Sadiku. Editora McGraw-Hill.
2. Circuitos elétricos. James W. Nilsson, Susan A. Riedel. Editora: Pearson; Edição: 8.
3. Microeletrônica - 5ª Ed. - Volume Único (Cód: 1970232). Sedra,Adel S. Editora Pearson.
4. Fundamentals of Microelectronics. Behzad Razavi. Editora John Wiley & Sons; Edição: 2nd Edition (24 de dezembro de 2012).

Abraços e até a próxima!

Editada por : Vitor Vidal
Engenheiro eletricista, mestrando em eng. elétrica e apaixonado por eletrônica, literatura, tecnologia e ciência. Divide o tempo entre pesquisas na área de sistemas de controle, desenvolvimento de projetos eletrônicos e sua estante de livros.

Revisada e editada por : Gustavo Murta
Técnico em eletrônica, formado em Curso superior de TPD, pós-graduado em Marketing. Trabalhou por muitos anos na IBM na área de manutenção de computadores de grande porte. Aposentou-se, podendo curtir o que mais gosta : estudar e ensinar Tecnologia. Hobbista em eletrônica desde 1976. Gosta muito de Fotografia e Observação de aves.

Versão 2.0 – agosto de 2018

APOSTILA KIT

ARDUINO INTERMEDIATE

Esta apostila acompanha o **Kit ARDUINO INTERMEDIATE** da Eletrogate, e contém conteúdos relacionados a todos os componentes do Kit.

WWW.ELETROGATE.COM